UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

Ingeniería Técnica en Telecomunicación Telemática

PROYECTO FIN DE CARRERA

AUTOMATIZACIÓN DEL CONTROL DE INVENTARIO MEDIANTE REDES DE SENSORES

Autor: Pablo Romaus Campos
Tutor: Ignacio Soto Campos

Leganés, Octubre de 2015

Título: AUTOMATIZACIÓN DEL CONTROL DE INVENTARIO MEDIANTE REDES DE SENSORES
Autor: Pablo Romaus Campos
Director: Ignacio Soto

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Llegar aquí no es solo mérito mío, y no lo digo solamente porque se me brinde la posibilidad de otorgar agradecimientos. Llegar aquí es fruto de unos Padres incansables que han dado todo de lo que disponían para permitirme estudiar y terminar mi carrera, es mérito de una novia, cariñosa y divertida que ha hecho de tripas corazón y ha convertido mi ausencia en una forma nueva de conectar y conocernos, es mérito de una hermana que ha visto siempre en mí más de lo que las pruebas tangibles y los resultados mostraban, llegar aquí es mérito de un tutor que nunca ha dejado de brindarme su consejo y apoyo para poder terminar a pesar de haber tenido que batallar con mis largas ausencias, es mérito de muchos compañeros y amigos de prácticas que han podido dar el extra que a mí me ha faltado en muchas ocasiones, es mérito de aquellos que me dieron un trabajo cuando no lo merecía y eso me ha permitió amar mi profesión de un modo nuevo.

Por ello Pedro Romaus, Matilde Campos, Vanessa Romaus, Carmen Gallego, Ignacio Soto quiero dejar vuestros nombres aquí pues realmente formáis parte de esto más de lo que creéis. Cada línea de código, cada frase, cada día agotador que esto me superaba, un abrazo, unas palabras de ánimo, un mail de apoyo, me permitieron seguir un poco más y terminar una función, un párrafo o un esquema hasta poder terminar. Por eso mil gracias a todos.

Resumen

El propósito del presente proyecto consiste en el desarrollo de una aplicación de control de inventario. Este sistema se compone de una red de nodos de sensores (Waspmote) conectados en modo infraestructura contra un punto de acceso de red WLAN. El back-end es una aplicación web ejecutándose sobre el contenedor de servlets Apache Tomcat.

Waspmote es un dispositivo ampliamente configurable mediante distintos tipos de módulos y sensores. La configuración hardware que se ha utilizado en los nodos Waspmote está conformada a por un módulo WiFi, como dispositivo de comunicación de datos, y un módulo RFID, para la lectura de las tarjetas de los elementos a inventariar, ambos conectados a la placa madre del dispositivo Waspmote.

Los nodos se encuentran sincronizados con el servidor del sistema utilizando el protocolo NTP. Para la transmisión hacia el servidor de los datos leídos haciendo uso del módulo RFID, los dispositivos Waspmote se valen del protocolo web HTTP 1.0. Cuando se produce algún error en la transmisión de los datos, los nodos disponen de una tarjeta SD que es utilizada para almacenar y guardar las cadenas HTTP que iban a ser enviadas al servidor.

Desde el punto de vista de los usuarios administradores, la aplicación de control de inventario actúa como una interfaz web para los datos de la lógica del sistema. Esta interfaz cuenta con diferentes roles de acceso que proporcionan a los administradores diferentes capacidades de actuación sobre el sistema.

La solución utiliza también una aplicación DNS y un servidor SMTP que complementan las funcionalidades del sistema de control de inventario.

Palabras clave: Inventario, Waspmote, WiFi, RFID, NTP, HTTP, SD, web, administración, DNS, SMTP

Abstract

The aim of this project is the development of an inventory control application. This system consists of a network of nodes connected in infrastructure mode against a WLAN access point device. The back-end is a Web application running on the servlet container Apache Tomcat.

Waspmote is a highly configurable device that allows using different types of modules and sensors. The hardware configuration used in the Waspmote node is formed by a WiFi module as communication device, and a RFID module for reading the items to inventory, both connected to the motherboard of the Waspmote device.

These nodes are synchronized with the system server using the NTP protocol. To transmit the data read using RFID module to the server, the Waspmote devices use the web protocol HTTP 1.0. When an error occurs in the transmission of the data, the nodes have an SD card that is used to store and save the HTTP chains that were going to be sent to the server.

From the point of view of the administrator users, the inventory control application acts as a web interface for the system logic data. This interface has different access roles that grant to the administrators different capacities for action on the system.

The solution also has installed a DNS application and a SMTP server which complement the functionality of the inventory control system simulating a real business environment.

Keywords: Inventory, Waspmote, WiFi, RFID, NTP, HTTP, SD, web, administration, DNS, SMTP.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

Algunos de los avances en la tecnología de la última década están propiciando el surgimiento de nuevos dispositivos más baratos, más rápidos y con nuevas capacidades de comunicación inalámbricas. El escenario que parece estarse describiendo con estas nuevas tecnologías resulta muy cercano a la idea descrita por el investigador Kevin Ashton en 1999 con el término, "Internet of Things". En ella se exponía que la mayoría de la información disponible en internet en aquel momento había sido generada por acción humana. Con "Internet of Things" se hablaba de un futuro donde existirían dispositivos que serían capaces de generar contenido de manera autónoma. Se describía que esta capacidad sería posible al asociar estos dispositivos con lectores RFID, sensores de bajo consumo, cámaras y demás elementos.

Las ventajas de pensar en tener dispositivos constantemente conectados capaces de generar contenido automáticamente en un entorno donde todo estuviera conectado nos hace imaginar posibilidades de conocer información relevante de forma instantánea. Ejemplos sencillos son la temperatura a la que se encuentra un punto concreto en una extensión forestal con riesgo de incendio, la localización en tiempo real de un elemento en un sistema de control de inventario o el estado de un paciente que se encuentra bajo monitorización.

Finalmente parece ser que el actual despegar de las redes de sensores están dando forma a esta idea. Las redes de sensores están compuestas por nodos autónomos capaces de recolectar y generar enormes cantidades de datos en un tiempo relativamente corto y de

encaminarlos hacia un elemento de su entorno para su almacenamiento y procesado. Actualmente las aplicaciones sustentadas en redes sensores se extienden en una gran cantidad de entornos y hacen posible algunas de las posibilidades descritas en el párrafo anterior.

Este proyecto fin de carrera surge con la motivación inicial de experimentar con las capacidades de las redes de sensores en la automatización de sistemas de inventario haciendo uso de la tecnología RFID. Este documento expone el análisis de la tecnología a estudio, el diseño y el desarrollo de una aplicación de control de inventario.

1.2 Objetivos

El objetivo principal del proyecto es el de ganar experiencia en el desarrollo sobre el hardware Waspmote y con ello poder dar forma a un sistema de control de inventario. Es importante recalcar que entre los objetivos nunca ha estado desarrollar una solución completa de un sistema control de inventario. La aplicación de control de inventario es la excusa para ganar conocimientos y experimentar con el desarrollo del dispositivo Waspmote. En consonancia con este objetivo principal surgen de manera natural una serie de objetivos secundarios:

- Conocer las características de una red se sensores y sus posibles aplicaciones.
- Las posibilidades en la integración con RFID (Radio Frequency IDentification) de las redes de sensores.
- Diseñar una solución de un sistema de control de inventario que incluya la placa Waspmote en la misma.
- Desarrollar el sistema completo: El código del Waspmote, los requisitos marcados para la aplicación de control de inventario. Y el despliegue o desarrollo de todos los sistemas perimetrales que permitan dan forma a la solución final.

1.3 Medios empleados

Para el desarrollo del proyecto se ha hecho uso de una serie de dispositivos hardware y soluciones software para poder realizar todas las tareas que han sido necesarias durante las distintas fases del mismo.

En las tablas siguientes se muestran los medios que se han utilizado:

Recursos Hardware	
Hardware	**Uso**
PC Laboratorio Telemática	El primer acercamiento a la plataforma Waspmote se realizó sobre este equipo.
Sobremesa Packard Bell iMax	Sobre él se desplego el entorno de desarrollo.
Portátil Asus F555L	Sobre él se desplego el entorno de desarrollo definitivo.
WaspmoteProV1.2	Es el hardware a estudio y el que ha ejecutado el software de los nodos en el sistema.
Módulos WiFi para Waspmote	Es el módulo de comunicación a estudio escogido para la placa Waspmote.
Módulos RFID para Waspmote	Es el módulo RFID escogido para la integración en el sistema final.
Tarjetas RFID	Se han utilizado para realizar lecturas con el módulo RFID.
Keyring RFID	Se han utilizado para realizar lecturas con el módulo RFID.
Router wifi linksys wrt54g	Es el recurso de red de la universidad utilizado para generar una WLAN para este proyecto durante el tiempo de uso del PC del laboratorio de telemática.

Tabla 1. Listado de medios hardware utilizados.

Recursos Software		
Software	**Versión**	**Uso**
Waspmote Pro IDE	04	Es utilizado para desarrollar, compilar y subir el código a los dispositivos Waspmote.
Sublime Text	3	Es el editor utilizado para el desarrollo del código del servidor.
Java	1.7.0_51	Es la plataforma de desarrollo del entorno servidor
Apache Tomcat	7.0.26	Utilizado en la aplicación como servidor web y como contenedor de Servlets.
MySQL	5.5.43	Base de datos SQL del sistema.
VMware Player	7.1.0	Utilizado para virtualizar el entorno de desarrollo.
NTP daemon	4.2.6	Demonio NTP utilizado para sincronizar el entorno.
Bind9	9.8.1	Servidor DNS utilizado para simular un entorno real.
Postfix	2.9.6	Servidor de correo utilizado para simular un entorno real.
Microsoft Office	365 Pro Plus	Se ha utilizado para la redacción de este documento y para preparar la presentación.
Microsoft Visio	Professional 2013	Creación de diagramas y esquemas.
GanttProject	2.6.6	Utiliza para crear el diagrama de Gantt asociado al proyecto.

draw.io	5.0.5	Aplicación online para la creación de diagramas y esquemas.

Tabla 2. Listado de medios software utilizados.

1.4 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye en esta sección un breve resumen del contenido de cada uno de los capítulos que componen el documento del presente proyecto:

- **Estado del arte:** En este capítulo se expone brevemente el estado de las tecnologías del campo de estudio.
- **Tecnologías utilizadas:** El objetivo de este capítulo es presentar las herramientas que se han utilizado en el desarrollo del proyecto.
- **Análisis y Diseño:** Esta sección plantea el diseño del sistema completo partiendo de la topología escogida para el sistema y los requisitos pactados.
- **Desarrollo:** El capítulo de desarrollo contiene los detalles en la implementación del código y la configuración del resto de elementos necesarios para la realización del sistema de control de inventario objetivo.
- **Pruebas:** Se incluye una pequeñas batería de pruebas sobre la funcionalidad descrita en capítulos anteriores
- **Conclusiones y líneas de trabajo futuras:** Conclusiones del desarrollo del proyecto, de sus características y de la viabilidad de la solución escogida. Así como la inclusión de una pequeña lista de posibles ideas de desarrollo futuras.
- **Planificación y presupuesto:** En este capítulo se expone una planificación de las fases y tareas que ha implicado el desarrollo del proyecto. También se incluye una previsión de costes en la realización de las distintas fases del proyecto.

Capítulo 2

Estado del arte

En el presente capítulo se pretende mostrar un pequeño acercamiento a los orígenes, evolución y características más representativas del campo tecnológico donde se enmarca el presente proyecto. Y mostrar algunos ejemplos de aplicaciones comerciales que utilizan soluciones completas de control de inventario con WSNs.

2.1 Redes inalámbricas de sensores

En los puntos siguientes se muestra un resumen de los puntos más representativos de las WSN.

2.1.1 Historia de las redes inalámbricas de sensores

Podemos decir que la las Redes inalámbricas de Sensores, identificadas a partir de aquí como WSNs (Wirelles Sensor Networks), tienen un origen dual, puesto que se puede hablar de un temprano origen militar en plena guerra fría, de las que se obtiene como resultado redes de sensores ya funcionales, y varias décadas después descubrimos una investigación más formal por parte de la Defense Advanced Research Projects Agency (DARPA) que termina saltando al entorno civil por medio de universidades y la empresa privada.

El temprano origen militar se remonta la década de 1950 durante la Guerra Fría y se debe al desarrollo por parte del ejército de Estado Unidos del sistema Sound Surveillance System (SOSUS). El sistema consistía en un conjunto de sensores acústicos (hidrófonos) situados en el fondo del océano y desplegado en lugares estratégicos para detectar y rastrear submarinos soviéticos. Se podría decir que este sistema de vigilancia submarino es la primera red inalámbrica de sensores que recuerda a una WSN actual. SOSUS no fue el único sistema de sensores inalámbricos desarrollados y desplegados durante la Guerra Fría por el ejército estadounidense. Las redes de radares de defensa aérea para defender el territorio continental de Estados Unidos y Canadá también fueron desplegadas por entonces.

Estas redes de sensores generalmente adoptaban una estructura de procesamiento jerárquico, el procesamiento de la información ocurría a varios niveles de donde se originaba el evento percibido por los sensores. En muchos casos, el operador humano era una parte relevante del funcionamiento de estas WSN. A pesar de no ser el objetivo de estas investigaciones proporcionó algunas tecnologías de procesamiento que resultaron claves para el desarrollo de las actuales WSNs. [1]

La investigación moderna de las redes de sensores comenzó en 1980 cuando la Defense Advanced Research Projects Agency (DARPA) inició el programa Distributed Sensor Network (DSN) con el fin de investigar de manera formal los desafíos en la implementación de redes de sensores distribuidas. Por aquel entonces, ARPANET (predecesora de internet) se encontraba funcionando y ya disponía de 200 equipos pertenecientes a universidades y centros de investigación. El programa DSN pretendía investigar si el enfoque sobre el que se sustentaba ARPANET para la comunicación podría extenderse a redes de sensores. Para ello se asumió que la red tendría muchos nodos de sensores de bajo coste distribuidos que colaborarían entre sí, pero operarían de manera autónoma, y que la información sería enrutada al nodo que pudiera utilizar mejor lo datos. Se puede decir que este fue un programa muy ambicioso dado el estado de la tecnología por aquel entonces.

Con el nacimiento de DSNs, y tras la implicación del mundo académico en su investigación, las redes de sensores encontraron un entorno para su desarrollo en el campo civil, gobiernos y universidades empezaron a usar WSNs en aplicaciones tales como monitoreo de la calidad del aire, la detección de incendios forestales, la prevención de desastres naturales y aplicaciones relacionadas con la monitorización del clima. Poco a poco a medida que nuevos estudiantes se incorporaron a la industria privada se empezó a promover el uso de redes inalámbricas de sensores en aplicaciones industriales como la distribución de energía, tratamiento de aguas residuales y la automatización de procesos industriales. Pero, a medida que la demanda crecía, ir más allá de las aplicaciones mencionadas suponía un desafío puesto que, por aquel entonces, las WSNs se basaban en sensores caros y voluminosos, además estas WSNs utilizaban protocolos de red propietarios. Las WSNs hasta entonces tenían entre sus características prioritarias la funcionalidad y el rendimiento pero, todavía, no contaban con otras características que serán importantes para llegar a convertirse en la WSNs actuales, como la facilidad de despliegue, el reducido tamaño de los sensores, costes de hardware bajos, el uso de estándares de red o el reducido consumo de energía. El consumo de energía, los altos costes y el tamaño de los sensores fueron algunas de las razones que impidieron un despliegue mayor de esta tecnología en una gama más amplia de aplicaciones [1] [2] [3]

2.1.2 WSN en la actualidad

En la última década las WSN han sufrido un fuerte crecimiento motivado por mejoras tecnológicas que han propiciado el desarrollo de electrónica cada vez más precisa, potente, barata y eficiente energéticamente lo que ha supuesto una reducción significativa del coste por nodo. Las WSNs son una de las tecnologías con un futuro más prometedor por la gran cantidad de campos de aplicación que parecen tener.

Actualmente las WSNs se encuentran en una gran variedad de aplicaciones, como, aplicaciones militares, monitorización de entornos naturales para prevención de desastres, sistemas estructurales de edificios como climatización o sistema de alarmas, aplicaciones domóticas, monitorización de pacientes, y muchas más.

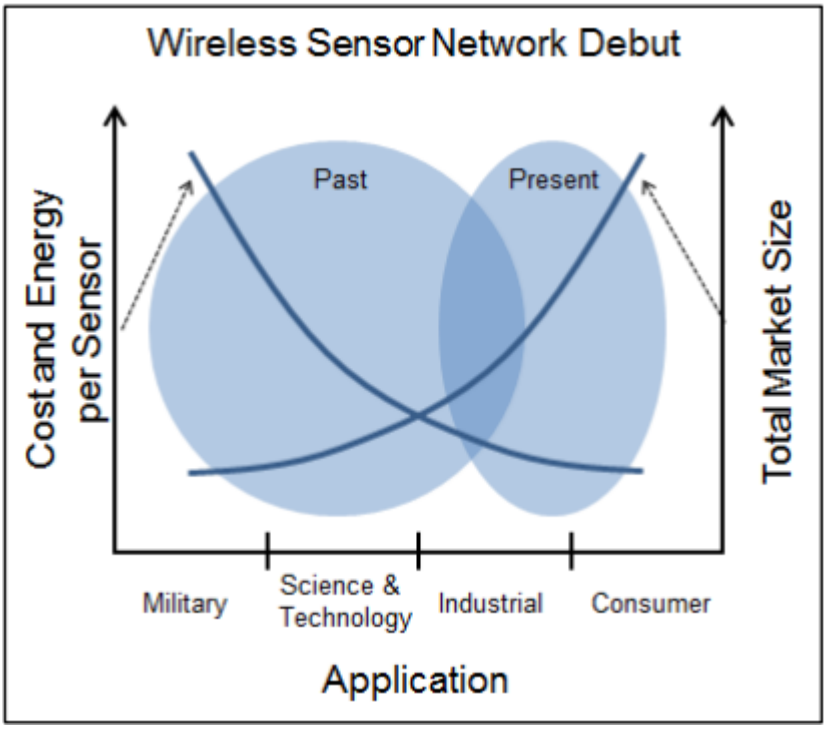

Figura 1. Relación entre el coste de los sensores y tamaño del mercado de las WSNs [3]

2.1.2.1 Generalidades de una WSN

Dada la gran cantidad de aplicaciones y campos en las que WSNs tienen cabida no existe una única lista de características que dan cabida a la definición de una WSN, pero aun así se puede enumerar una lista de puntos que deben ser abordados de manera cuidadosa en todos los casos en que se trabaja en el diseño o despliegue de una WSN:

- **Autonomía:** Podemos hablar de las necesidades de autonomía en un nodo de una WSN en los siguientes puntos:

    - Energética: es una característica esencial para los nodos de una WSN su capacidad de administrar de manera eficiente este recurso tan vital ha sido motivo de estudio durante las décadas pasadas [4] [5] [6] [7]. De estos estudios se deduce el peso que tiene el algoritmo de enrutado y la jerarquía de la topología en la eficiencia energética.

      Las comunicaciones son la mayor fuente de consumo energético en los nodos de una WSN. Algunas soluciones como algoritmos de enrutado basados en cluster parecen mejorar la autonomía energética de los nodos de una WSN. En el punto 2.1.2.2 se cuenta en qué consisten las WSN con enrutado basado en cluster.

    - Comunicaciones inalámbricas: Los nodos de una WSNs deben de tener la capacidad de poder reconfigurar sus comunicaciones de manera autónoma por cambios en su entorno, como movimientos de sensores en la red o adición o desaparición de nodos vecinos.

- **Coste de los nodos:** La vida útil de un nodo se basa en piezas hardware como su batería, sus interfaces de red, los sensores de los que dispone, susceptibles de fallar. Las WSNs en muchas ocasiones se encuentran en entornos hostiles o de difícil acceso y la sustitución de nodos o elementos de hardware no es posible debido a las limitaciones físicas. Por lo que en algunas aplicaciones se habla de sensores de un solo uso, ya que una vez desplegados se asume que no son recuperables, por lo que, esto convierte al factor coste en una característica a tener en cuenta a la ahora de desplegar una WSNs. Además el factor coste es una de las razones principales de la evolución de las WSNs.

- **Escalabilidad:** La arquitectura y protocolos de redes de sensores deben ser capaces de ampliarse hasta cualquier número de sensores. Normalmente las WSNs consisten en un gran número de sensores distribuidos por una amplia área geográfica, el número de sensores puede llegar a ser del orden de cientos e incluso miles y el rendimiento de la red no debería verse mermado.

  Aun así en el diseño de una WSN es necesario tener presente la topología empleada. Las topologías con un solo salto de enrutado son por su naturaleza más escalables que las topologías multisalto, puesto que, en estas últimas, un incremento de los nodos puede ocasionar un aumento de los saltos hacia el elemento que procesa los datos.

- **Redes centradas en contenido:** El objetivo de una red de comunicación es el de proporcionar conectividad entre los distintos elementos de la red, el de transportar un conjunto de bits de un punto a otro de la red, en cambio aunque el necesidad de conectividad todavía existe, no es el objetivo de una WSN. Las WSNs existen para poder proporcionar información, no simplemente transportarla. Su fin es la recolección de datos y su entrega hacia el elemento con la capacidad de procesar la información.

2.1.2.2 Protocolos de enrutado

Aunque el análisis en profundidad de los protocolos de enrutado queda fuera del alcance de este proyecto hemos considerado interesante mostrar un pequeño resumen. El enrutado de una WSN ha sido estudiado en una gran cantidad de documentos con el fin de mejorar algunas de las características de las WSN, principalmente, el consumo y la escalabilidad En algunos estudios se ha presentado una organización de las características de los protocolos de enrutado de una WSN (ver Figura 2). En esta propuesta se organizan las características de los protocolos según la estructura de la red de la WSN o según la operación del propio protocolo [8] [9].

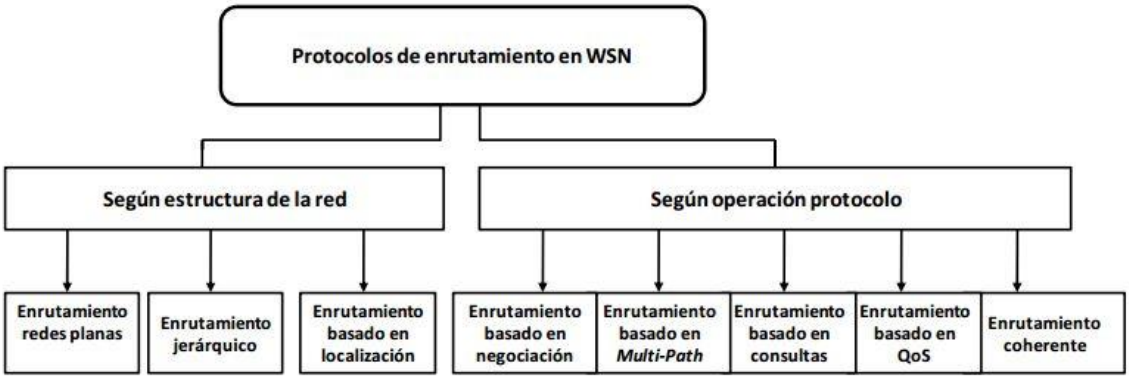

Figura 2. Clasificación de los protocolos de enrutado de una WSN [9]

En la arquitectura común de una WSN nos encontramos con un número elevado de nodos sensores con capacidades de comunicación inalámbricas desplegados en un terreno de detección. Los datos captados por los nodos sensores son enviados hacia uno o varios nodos sumidero con capacidades de acceso a internet o a la red donde se encuentra el servidor o elemento con capacidades de gestión de la información. La naturaleza de la topología de red existente entre los nodos es variable y nos podemos encontrar con:

- **Redes planas (Flat Networks)**: Son redes donde todos los nodos tienen la capacidad de enrutar sus datos y los de sus vecinos hacia el nodo sumidero, denominadas como redes planas puesto que todos los nodos son homogéneos y realizan las mismas labores de enrutamiento y procesamiento.

- **Redes Jerárquicas (Clustered Networks)**: En este tipo de topología existen una heterogeneidad entre los nodos y no todos desempeñan las mismas funciones, existen diferentes roles. En esta clase de redes los nodos se asocian en topologías jerárquicas llamadas Clúster. Los nodos con un suministro de energía mayor son utilizados para el procesamiento y enrutado de los datos hacia el nodo sumidero, se les conoce como Cluster Heads (CH), y los nodos de un suministro energético menor son relegados a labores de recabar datos por medio de sus sensores Como se ha comentado antes este tipo de soluciones suponen una mejora en la eficiencia energética pero también contribuye en la mejora del tiempo de vida y la escalabilidad del sistema.

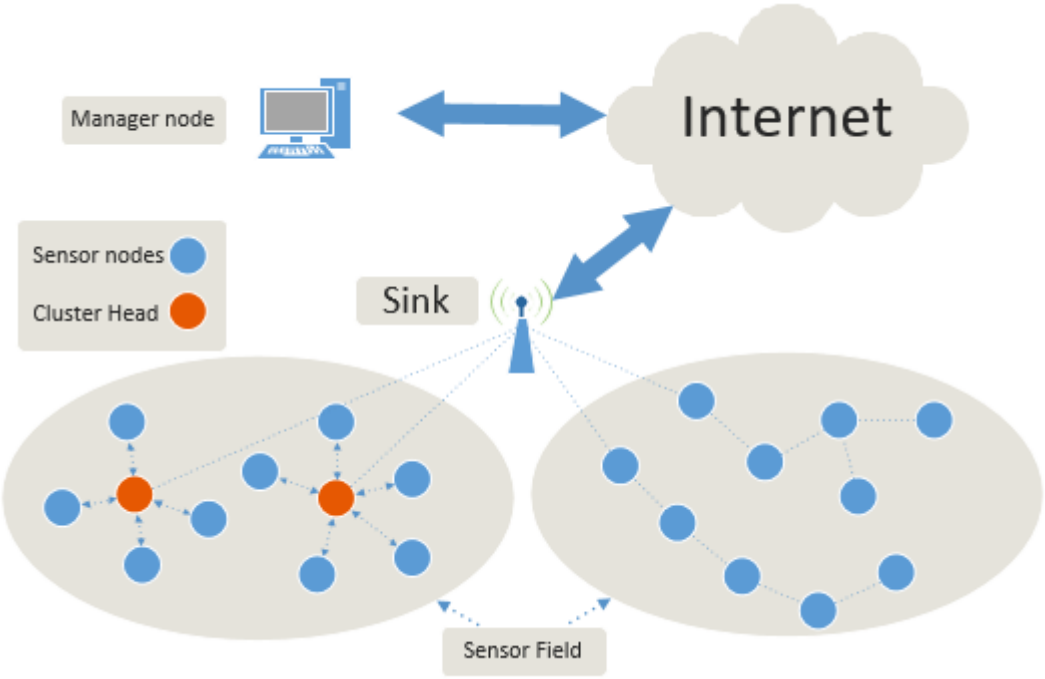

Figura 3. Diagrama de partes en la topología de un WSN [9]

- **Redes basadas en localización**: En este tipo de encaminamiento las redes son direccionadas por la localización de los nodos.

  Para ello pueden valerse de métodos como obtener las coordenadas relativas de los nodos vecinos por interpolación con los nodos vecinos determinando la distancia entre nodos por el cálculo de la intensidad de la señal de entrada o utilizar módulos GPS de bajo coste en los nodos.

Siguiendo la clasificación de los protocolos de enrutado según su modo de operación de los protocolos de enrutado nos encontramos con las siguientes tipos de enrutado:

- **Basado en negociación:** Estos protocolos pretenden eliminar el envío de datos redundantes al nodo anexo para su enrutado por medio de una serie de mensajes de negociación antes del envío de los datos reales. Esto evita la inundación o el solapamiento de envíos de datos redundantes.

- **Multicamino (Multipath):** Estos protocolos utilizan varios caminos de enrutado en lugar de unos solo. Esto mejora el nivel de tolerancia a fallos de la red.

- **Basado en consultas:** En este tipo de enrutado, el elemento de la red con capacidad de procesar la información envía una consulta solicitando unos datos. El nodo que posee los datos se los envía al remitente de la consulta.

- **Basado en QoS (Quality of Service):** Estos protocolos de enrutado se caracterizan por cumplir un parámetro de calidad de servicio (retardo, ancho de banda,…) y balancear los recursos del nodo (consumo de energía) para cumplir con esa variable.

- **Coherente:** En los protocolos de enrutado coherente los datos son mínimamente tratados (marcas de tiempo, datos duplicados) antes de ser pasados a otro nodo para su enrutado. Si un protocolo de enrutado es no-coherente los nodos pueden hacer un tratamiento de los datos más intenso antes de pasarlo a otro nodo para su enrutado, esto incrementa el consumo del nodo.

Es importante señalar la idea de que algunos protocolos pueden ser encuadrados al mismo tiempo en varias de las categorías analizadas en esta sección (las categorías no son excluyentes).

2.1.2.3 Partes de un nodo sensor

En general, los nodos sensores se componen de un hardware muy simple formado por (ver Figura 4):

- Una unidad de proceso: Conformada por un procesador embebido, generalmente un microcontrolador y la memoria.

- Una unidad sensora: Que está compuesta por uno o varios sensores y un conversor analógico a digital.

- Una unidad de transmisión: Esta unidad se encarga de las comunicaciones inalámbricas del nodo. Está compuesto por transceptor capaz de enviar y recibir datos.

- Un módulo de energía: Que se encarga de alimentar todos los módulos que componen el hardware del nodo. Normalmente el sistema dispone de una batería utilizada para almacenar la energía.

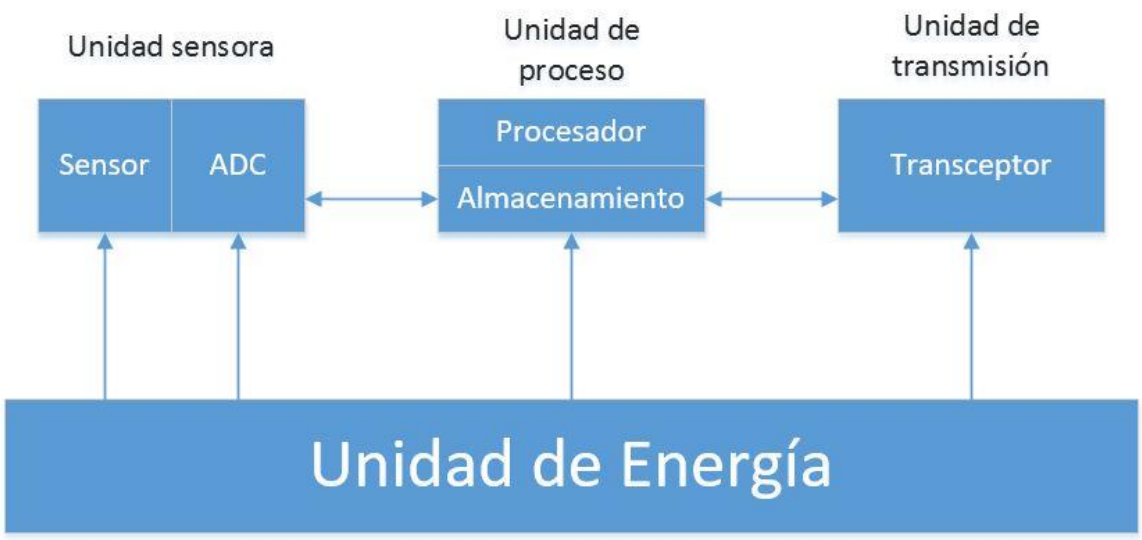

Figura 4. Componentes de un nodo Sensor

2.2 RFID y redes inalámbricas de sensores

En los puntos siguientes se analizan brevemente las características de RFID y sus posibilidades en la WSNs. Terminaremos analizando su uso en sistemas de control de inventario.

2.2.1 Tecnología RFID

RFID es una tecnología inalámbrica de transmisión de datos usada fundamentalmente para el seguimiento de la localización de etiquetas o la identificación automática. Un sistema RFID se compone dos elementos diferenciados:

- **Elemento transpondedor:** El elemento transpondedor o tarjeta RFID dispone de una antena y un chip. La antena permite al chip enviar los datos que tiene almacenados.

- **Elemento interrogador:** El elemento interrogador o lector RFID dispone de una antena que genera a su alrededor un campo electromagnético.

Cuando la tarjeta RFID entra en el campo electromagnético generada por la antena del lector, la corriente inducida en la tarjeta permite al chip enviar los datos encapsulados desde la tarjeta RFID al lector RFID. Por su parte el lector detecta la presencia de la tarjeta como una perturbación del nivel de señal.

2.2.2 Integración de RFID en WSNs

En referencia a todo lo comentado hasta ahora podemos decir que las WSNs son usadas para obtener información por medio de sensores de su entorno mientras que la tecnología RFID es usada en aplicaciones de identificación y detección.

La idea de la integración de RFID en una red de sensores pasa por la incorporación del elemento interrogador de un sistema RFID en un nodo sensor de WSN. El microcontrolador del nodo se encargaría también de gestionar el lector RFID.

La integración de ambas tecnologías permitiría a un sistema RFID cubrir una gran extensión de terreno y utilizar los nodos sensores vecinos de la red inalámbrica para que los datos adquiridos por el lector RFID sean enrutados hacia el servidor que se encargue de gestionar dichos datos.

Algunos ejemplos de campos en los que la integración de ambas tecnologías podría ser utilizada seria en el entorno militar, en de la salud, en hogares inteligentes o la gestión de almacenes. [10]

2.2.3 Sistemas de gestión de almacenes

La necesidad de un control más preciso en la gestión de almacenes ha visto en RFID una posibilidad de mejorar esta carencia. A continuación analizamos brevemente algunos proyectos y estudios llevados a cabo con RFID y redes de sensores inalámbricas para la gestión de almacenes:

- **RFID-IWMS (RFID-based intelligent warehouse management system)** [11]**:** En este proyecto se describe el reto de la integración RFID en un WMS (warehouse management system).

  Este proyecto incluye el empleo y adaptación del siguiente hardware para adaptarlo al conjunto del sistema:

  - Terminales RFID que permiten leer y escribir tarjetas RFID.
  - Carretillas elevadoras con lectores RFID y antenas WLAN.

  La Figura 5 muestra un esquema de la estructura utilizada en el desarrollo del proyecto que describe el documento

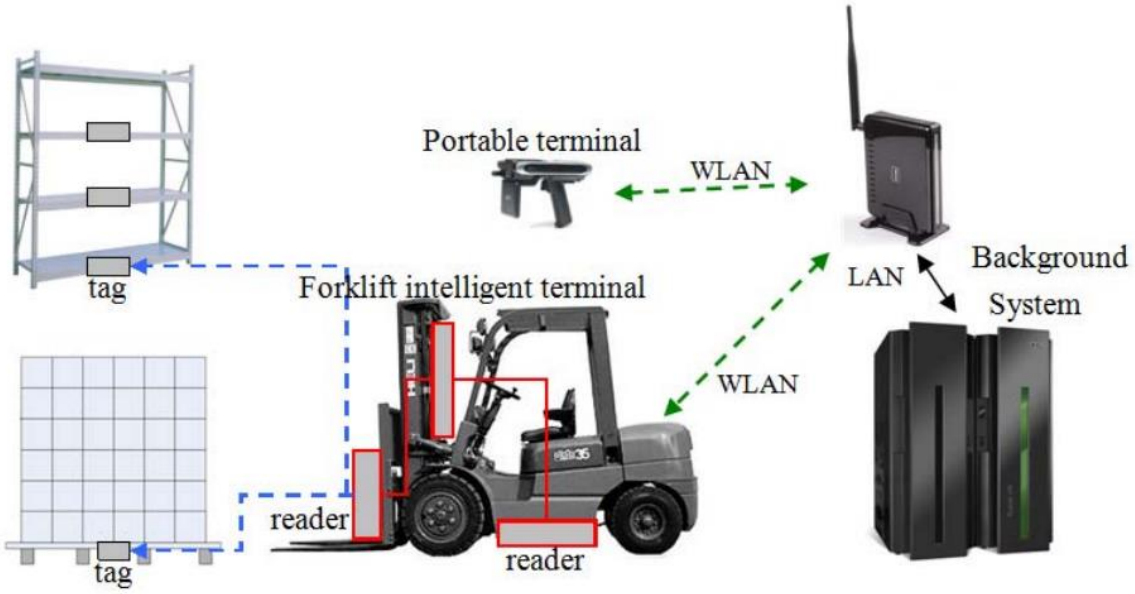

Figura 5. Estructura del sistema utilizado en RFID-IWMS (tomada de [11])

- **"An Automatic RFID and Wireless Sensing System on a GHS-based Hazardous Chemicals Management Platform"** [12]**:** Este es un estudio describe la integración de RFID y tecnologías inalámbricas, para la comunicación de los dispositivos, en un sistema de gestión de productos químicos peligrosos. La incorporación de RFID a este sistema ha permitido realizar inventarios automáticos periódicos y minimizar la interacción humana en el sistema.

  En esta solución se ha incorporado un dispositivo RFID y un sistema de comunicación inalámbrica a una célula de toma de pesos. En la topología existe un dispositivo maestro y una serie de elementos esclavo. El elemento maestro enruta los datos de los elementos esclavos hacia el servidor Figura 6.

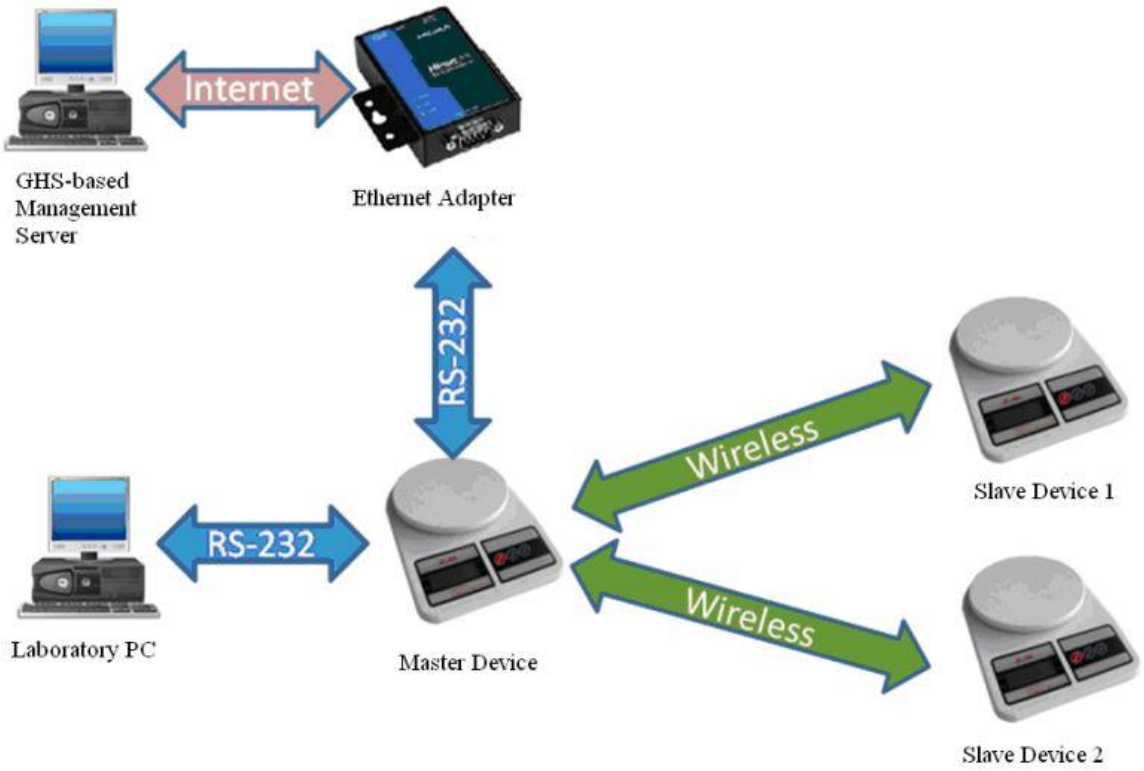

Figura 6. Estructura del sistema utilizado en "GHS-based Hazardous Chemicals Management Platform" (tomada de [12]).

2.3 Conclusiones del capítulo

De cara al desarrollo del presente proyecto el actual capítulo nos ha permitido conocer el origen, el entorno y las características fundamentales de las WSNs. Los campos en los que esta tecnología tiene un presente y un futuro, las razones que han propiciado su investigación inicial, su más reciente éxito y su integración con RFID en la gestión de almacenes.

Si bien la naturaleza del proyecto no es una investigación exhaustiva de las WSN, gran parte de la información aquí descrita ha sido de utilidad en la realización de todo el proyecto, sobre todo, en la fase de diseño del mismo.

Capítulo 3

Tecnologías utilizadas

A continuación se analizan las tecnologías y herramientas más representativas que han sido utilizadas para la resolución del proyecto.

3.1 Waspmote

Waspmote es una plataforma orientada a desarrolladores que permite el despliegue y desarrollo de WSNs. Waspmote es una placa madre que destaca frente a otras soluciones similares para el desarrollo de WSNs por características como su versatilidad y sencillez. A continuación se extienden estas dos características:

- **Versatilidad:** Waspmote es una placa con una gran adaptabilidad. La placa Waspmote ofrece un gran número distinto de posibilidades de configuración hardware por medio de diferentes módulos y sensores que pueden acoplarse a la placa madre y le permiten dar solución a una gran cantidad de aplicaciones.

  A parte de todas las que se incluyen como parte de la solución del producto se ha permitido la posibilidad a los desarrolladores de incluir nuevas placas de sensores a Waspmote, esta posibilidad está contemplada en la API y en la documentación de Waspmote.

- **Sencillez:** Una de las ventajas de trabajar con Waspmote es sin duda la API de la placa Waspmote que abstrae del desarrollo a bajo nivel de este tipo de entornos y simplifica significativamente el desarrollo. El código de la API de Waspmote se encuentra totalmente disponible y documentado para desarrolladores.

Para el desarrollo del proyecto se ha trabajado en profundidad con la parte de la API que aplica a los módulos WiFi y RFID. La configuración hardware utilizada en el desarrollo del proyecto se compone de:

- Placa Waspmote Pro v1.2.
- Módulo WiFi con una antena de 2dBi.
- Módulo RFID/NFC 13.56MHz.
- Tarjeta SD.
- Placa de expansión para Radio Socket 1.
- Batería recargable con voltaje de 3.7V y 6600mAh Li-Ion

3.1.1 Aspectos Hardware

La localización física en la placa de todos los componentes hardware que serán descritos en los puntos sucesivos se pueden observar en la Figura 7 a continuación.

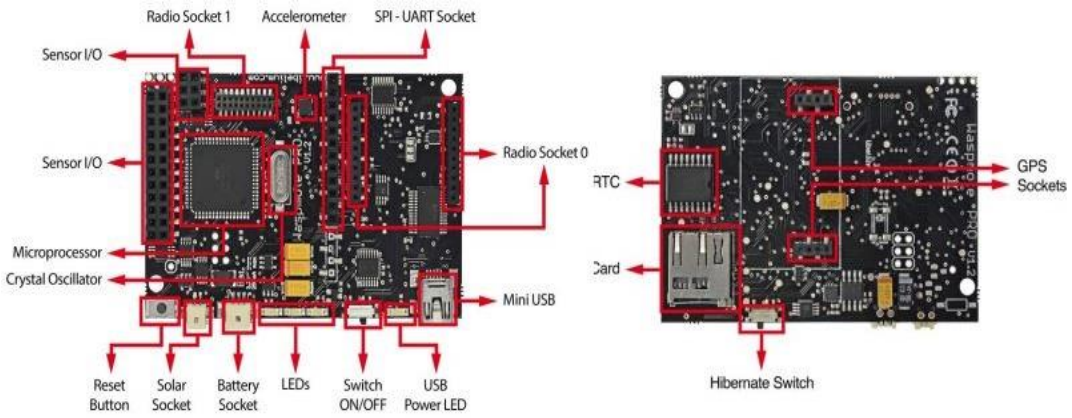

Figura 7. Componentes hardware de Waspmote, vista superior (izquierda) y vista inferior (derecha) [13]

El hardware de la placa Waspmote dispone en la parte superior de la placa de dos Radio Socket. Los pines de estos socket se utilizan para conectar a la placa madre los diferentes chipset de comunicaciones preparado para el uso con la placa Waspmote. Para el uso de la conexión de interfaces de red en el Radio Socket 1 o el uso simultáneo de ambos Radio Socket es necesario el uso de un módulo especial llamado Expansión Radio Board que adapta los pines del Radio Socket 1 a la forma del chipset de la radio.

3.1.1.1 Datos Generales

A continuación se describen las generalidades hardware relativas a la placa Waspmote y a su microcontrolador que se muestran en la Tabla 3.

Datos Generales	
Microcontrolador	ATmega1281
Frecuencia	14.7456MHz
SRAM	8KB
EEPROM	4KB
FLASH	128KB
Tarjeta SD	2GB
Peso	20gr
Dimensiones	73.5x51x13mm
Rango de Temperatura	[-10ºC,+65ºC+]
Reloj	RTC (32KHz)

Tabla 3.Generalidades de la placa Waspmote. [13] [14]

- **Microcontrolador**: El ATmega1281 es un microcontrolador de bajo consumo de 8-bit. La arquitectura del microcontrolador es RISC y dispone de:
    - 135 Instrucciones, la mayoría de ejecución de un solo ciclo.
    - 32 Registros de 8 bit de propósito general.
    - Un rendimiento máximo de 1MIPS por MHZ.
    - Además cuenta con las memorias SRAM de 8KB, EEPROM de 4KB y 128KB Flash.

- **Memorias:** La placa dispone de diferentes soluciones de almacenamiento.
    - **SRAM:** Es una memoria volátil de 8KB. Esta SRAM es compartida entre las variables (inicializadas y no inicializadas), la asignación dinámica de memoria y la pila de llamada a subrutinas.
    - **EEPROM:** Es una memoria persistente de 4KB que permite el almacenamiento de valores cuando la placa se encuentra apagada o en el estado “Hibernate” (el estado “Hibernate" se describe en el punto 3.1.1.2). Solo 3KB se encuentran disponibles para desarrolladores puesto que las posiciones de memoria 0 a1023 se encuentran reservadas para el almacenamiento de datos de fábrica.
    - **Flash:** En la memoria Flash de128KB del microcontrolador se encuentran tanto el programa que se ha subido a la placa y el programa de arranque de la placa. El uso del IDE de Waspmote evita sobrescribir el programa de arranque al cargar programas en la placa.
    - **SDCard:** La placa Waspmote soporta tarjetas SD hasta un máximo de 2GB de capacidad y con un sistema de ficheros FAT16.

- **RTC (Real Time Clock**): El modelo del reloj implantado en Waspmote es el DS3231SN. Es uno de los relojes más precisos del mercado. La mayoría de RTCs en el mercado tienen una variación de 1,7 seg de pérdida de precisión por día mientras que el modelo incorporado en la placa experimenta pérdidas de 0,16 seg por día.

3.1.1.2 Modos de Operación e Interrupciones

Las interrupciones permiten al microcontrolador alterar el flujo de ejecución natural de su programa para atender el evento que ha generado una interrupción. En el caso de Waspmote las interrupciones están muy relacionadas con los modos de operación, puesto que, son estas las que permiten los cambios de estado entre los mismos Figura 8.

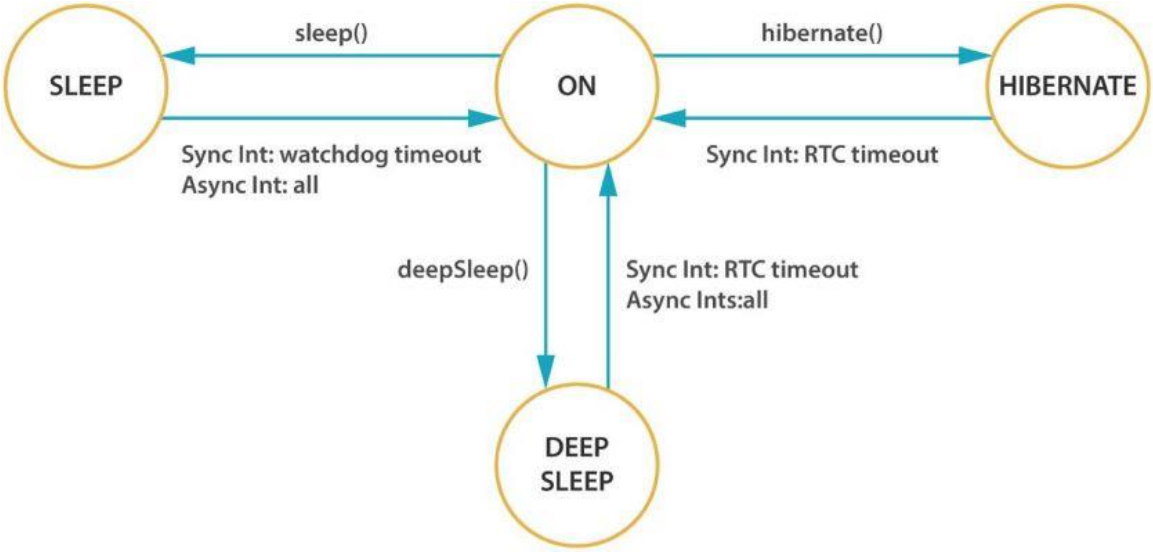

Figura 8. Relación entre los modos de operación y las interrupciones. [13]

Waspmote dispone de varias fuentes de interrupciones tanto síncronas como asíncronas:

- Asíncronas:
  - Sensores: umbral programable.
  - Acelerómetro: caída libre, impacto (umbral programable).
  - XBee (DigiMesh)

- Síncrona:
  - Watchdog: alarma programable de 32 ms a 8 s
  - RTC: alarma programable de 1s a días

Los distintos modos de operación varían el consumo de la placa Waspmote de un modo significativo. Como observamos en la Tabla 4 Waspmote tiene 4 modos de operación. A continuación se exponen algunos aspectos de estos modos:

Consumo	
ON	15mA
Sleep	55µA
Deep Sleep	55µA
Hibernate	0.07µA
Operativo sin recarga	1 año *usando el modo Hibernate*

Tabla 4.Consumos de la placa Waspmote en sus distintos modo de operación. [13] [14]

- **ON:** El modo normal de operación es, como se debería esperar, el modo con el consumo más alto de todos. Este modo acepta interrupciones tanto síncronas como asíncronas.

- **Sleep:** En este modo la ejecución del programa es pausada y la placa pasa a un estado donde el consumo disminuye hasta los 55 µA del que puede ser despertado por las interrupciones asíncronas y las interrupciones síncronas del Watchdog. El intervalo de tiempo configurable para el ciclo de este modo se encuentra entre 32ms a 8 s que es el intervalo de configuración que permite el Watchdog.

- **Deep Sleep:** Este modo es similar al modo Sleep pero en este estado solo las interrupciones síncronas del RTC y las asíncronas pueden despertar a la placa Waspmote de este estado. El intervalo de este modo varía de segundos a minutos, horas o días.

- **Hibernate:** En este estado el programa se para y la placa y todos los módulos se apagan, solo el RTC permanece conectado. Solo la interrupción síncrona del RTC puede despertar a la placa Waspmote del estado de "Hibernate". En este estado el consumo cae hasta los 0.07 µA.

3.1.1.3 Entradas y salidas

En la Tabla 5 podemos ver el número de entradas y salidas de la placa ordenada por tipos.

Entradas/Salidas	
Analog (I)	7
Digital (I/O)	8
PWM	1
UART	2
I2C	1
USB	1
SPI	1

Tabla 5. Número de entradas y Salidas de la placa Waspmote. [13] [14]

En el diagrama de la Figura 10 podemos ver la relación de las entradas y salidas con los elementos periféricos de la placa:

- **Pines Analógicos/Digitales**: Tanto las entradas y salidas digitales como las entradas analógicas se encuentran en los conectores marcados como "Sensor I/O" en la Figura 7.

- **UART (Universal Asynchronous Receiver-Transmitter):** Waspmote dispone de 2 UART según los datos listados en la Tabla 5:

  - UART0 es compartida por el puerto USB y el Socket0. El multiplexor controla la señal de entrada a la UART. Por defecto, la entrada seleccionada es el Socket0.

  - UART1 es compartida por 4 puertos:

    - UART1 AUX
    - UART2 AUX
    - Socket1
    - GPS

    De nuevo esta UART utiliza un multiplexor para controlar el acceso de los diferentes puertos que se comunican a través de ella. En el caso de esta UART es posible seleccionar cuál es el puerto que utiliza la UART desde el código.

```
{
  Utils.setMuxAux1(); // set Auxiliar1 socket
  Utils.setMuxAux2(); // set Auxiliar2 socket
  Utils.setMuxGPS();  // set GPS socket
  Utils.setMuxSocket1(); // set Socket1
}
```

Figura 9. Ejemplo de código de selección del multiplexor de UART1. [14]

- **I2C:** I2C es un bus de comunicaciones serie multi-maestro y multi-esclavo. Según la Figura 10 existen 2 periféricos en la placa Waspmote que comparten en paralelo la entrada I2C el acelerómetro y el RTC. En el caso de las comunicaciones I2C el microcontrolador siempre actúa como maestro y los dispositivos como esclavos.

- **SPI (Serial Peripheral Interface):** La entrada es utilizado por la tarjeta SD, el conector SPI/UART de la placa.

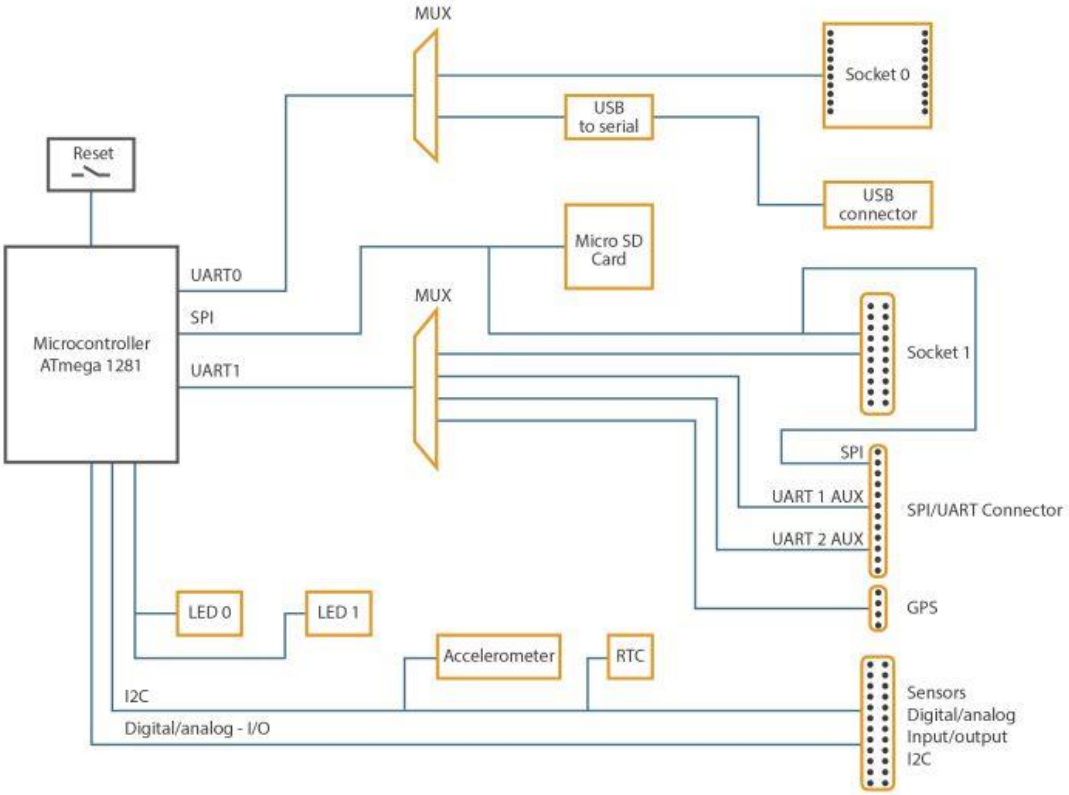

Figura 10. Diagrama de Señales de la placa Waspmote 1.2 Pro. [14]

3.1.1.4 Sensores en placa

Sensores montados en placa		
Temperatura	[-40ºC,+85ºC] Precisión: 0.25ºC	
Acelerómetro	±2g/±4g/±8g	
	Modos de alimentación	***Velocidad de salida de datos (Hz)***
	Power down	--
	Modo normal	1000
	Low-power1	0.5
	Low-power2	1
	Low-power3	2
	Low-power4	5
	Low-power5	10

Tabla 6. Características Técnicas de la placa Waspmote 1.2 Pro. [13] [14]

Waspmote cuenta con 2 sensores integrados en su placa madre, un sensor de temperatura y un acelerómetro.

- El sensor de temperatura forma parte del RTC y es usado por este para recalibrarse a sí mismo. Este sensor es capaz de medir temperaturas en el rango de -40ºC a +85ºC.

- El modelo del acelerómetro es LIS3331LDH es capaz de detectar la variaciones de aceleración en los 3 ejes del espacio (X, Y, Z). Como vemos en la Figura 10 este dispositivo se comunica con el microcontrolador por medio del interfaz I2C

3.1.1.5 Módulos de red

Waspmote dispone de una gran gama de interfaces de red diferentes que se adaptan a las distintas situaciones a las que se puede enfrentar una WSN. A continuación se describe el hardware de red disponible actualmente para el uso con la placa Waspmote:

- **802.15.4/ZigBee:** La tabla contiene los módulos ZigBee existentes y sus características.

Protocol	Module	Frequency	txPower	Sensitivity	Channels	Range
802.15.4	PRO	2.405-2.465GHz	100mW	-100dBm	12	7000m
ZigBee-Pro	XBee-ZB-Pro	2.4-2.7GHz	50mW	-102dBm	14	7000m
RF	XBee 868	869.4-869.65MHz	315mW	-112dBm	1	12km
RF	XBee 900	902-928MHz	50mW	-100dBm	12	10Km

Tabla 7. Características técnicas de los distintos módulos ZigBee. [14]

Los módulos XBee-802.15.4 y XBee-900 pueden usar un firmware opcional, **DigiMesh**, y con él pueden crear redes malladas en lugar de topologías punto a punto.

ZigBee es una de las interfaces que por sus características mejor parece adaptarse a las WSN, bajo consumo, muy escalable e integrable en hardware sencillo y en Wasmote permite conseguir de manera sencilla topologías en estrella o redes jerárquicas

- **LoRa (Long Range):** La tecnología LoRa [15] es la que mejor rendimiento en distancia ofrece. Las características técnicas del módulo se muestran a continuación:

Module	Frequency	txPower	Sensitivity	Channels	Distance	Antenna
SX1272	863-870 MHz	14 dBm	-134 dBm	8	22+ Km (13.41+ miles)	0dBi
	902-928 MHz			13		4.5dBi

Tabla 8. Características técnicas del módulo LoRa. [14] [16]

- **Bluetooth:** Esta tecnología dispone de dos módulos diferentes que trabajan con versiones diferentes del estándar.

Módulo Bluetooth Low Energy (BLE)							
Protocol	**Model**	**Frequency**	**txPower**	**Sensitivity**	**Channels**	**Distance**	**Antenna**
BLE / Bluetooth Smart	BLE112	2.4GHz	[-23, +3dBm]	-103 dBm	37	100m	2dBi/5dBi

Tabla 9.Características técnicas del módulo BLE. [17]

Módulo Bluetooth Pro					
Protocol	**Frequency**	**txPower**	**Channels**	**Distance**	**Antenna**
Bluetooth v2.1 + EDR. Class 2	2.4GHz	[-27, +3dBm]	79	10-50m	2dBi

Tabla 10. Características técnicas del módulo Bluetooth Pro. [18]

- **Módulos para redes móviles celulares:** Se dispone de 3 módulos que implementan distintas tecnologías de telecomunicaciones móviles.

Module	**Model**	**Frequency**	**txPower**	**Sensitivity**	**Antenna**
GSM/GPRS	SIM900 (SIMCom)	850/900/1800/1900MHz	2W 850/900 1W 1800/1900	-109dBm	0dBi
GPRS+GPS	SIM928 (SIMCom)	850/900/1800/1900MHz	2W 850/900 1W 1800/1900	-109dBm	0dBi
3G+GPS	SIM5218E (SIMCom)	UMTS 2100/1900/900MHz GSM/EDGE 850/900/1800/1900 MHz	0,25W 900/1900/2100 2W GSM 850/900 1W DCS1800/PCS1900	-106dBm	0dBi

Tabla 11.Carateristicas técnicas de los distintos módulos de tecnologías móviles.

- **Industrial Protocols**. Los módulos RS-485 y RS232 pueden hacer uso de la librería de software Modbus. Las características técnicas de los módulos industriales se muestran a continuación.

Module	RS-485 / Modbus	RS-232 Serial / Modbus	CAN Bus
Standard	EIA RS-485	TIA-232-F	ISO 11898
Physical Media	Twisted pair	Single ended	Twisted pair
Connector	DB9	DB9	DB9
Network Topology	Point-to-point Multi-dropped Multi-point	Point-to-point	Multimaster
Maximum Devices	32 drivers or receivers	N.A.	N.A.
Mode of Operation	Differential signaling	Unbalanced	Differential signaling
Maximum Speed	460800 bps	115200 bps	125 to 1000 Kbps
Voltage Levels	-7 V to +12 V	-25...+25	0-5V
Mark(1)	Positive Voltages (B-A > +200 mV)	-5...-15	N.A.
Space(0)	Negative Voltages (B-A < -200 mV)	+5...+15	N.A.
Available Signals	Tx+/Rx+, Tx-/Rx- (Half Duplex) Tx+,Tx-,Rx+,Rx- (Full Duplex)	Full Duplex (Rx, TX)	Half Duplex
Available sockets in Waspmote	socket 0	socket 0 socket 1	socket 0

Tabla 12.Caracteríticas técnicas de los módulos industriales. [14]

Las tecnologías y los módulos WiFi y RFID son descritos con más detalle en los puntos 3.1.1.6 y 3.1.1.7

3.1.1.6 Módulo WiFi

WiFi es una certificación de la Wi-Fi Alliance. La certificación Wi-Fi garantiza el cumplimiento de los estándares de la tecnología IEEE 802.11 por parte de equipos. Comúnmente se utiliza el nombre WiFi para referirse a la tecnología. El estándar 802.11 especifica el control de acceso al medio (MAC) y el nivel físico de redes locales inalámbricas (WLAN).

3.1.1.6.1 Características del módulo WiFi

A continuación se analizan algunas características y detalles técnicos del módulo WiFi utilizado en Waspmote:

- **Características radio:** La Tabla 13 muestra los rangos de trabajo y las características teóricas del módulo. La Tabla 14 muestra el tiempo teórico de autenticación en una WLAN en función de la tecnología de encriptación. La Tabla 15 muestra el consumo del módulo WiFi en los diferentes estados de trabajo.

Protocol	Frequency	txPower	Sensitivity	Channels	Distance	Antenna
802.11b/g	2.4GHz	0dBm–12dBm	-83 dBm	13	50-100m (2dBi) (5dBi)	2dBi/
					300-500m	5dBi

Tabla 13. Características técnicas del módulo WiFi. [14]

Encryption	Time (seconds)
Open	3,997 s
WEP	4,077 s
WPA	4,176 s
WPA2	4,293 s

Tabla 14.Tiempo para unirse a AP con encriptación. [14]

State	Power Consumption
OFF	0 µA
Sleep	4 µA
ON	33 mA
Receiving Data	38 mA
Transmitting Data	38 mA
Scanning Access Points	34 mA

Tabla 15.Relación entre el estado y el consumo del módulo WIFI. [14]

- **Microcontrolador**: El chip del módulo WiFI es el modelo WiFly RN171 y consta de las siguientes características hardware [19]:
    - Memorias:
        - 8Mbit Flash Memory
        - 128 KB RAM
        - 2MB de ROM
        - 2 KB de memoria battery-backed
    - Procesador SPARC de 32- bit
    - Un RTC propio para marcas de tiempo, auto-sleep y auto-wakeup.
    - Un conversor analógico digital
    - Un dispositivo Crypto Accelerator para ayudar a la CPU
    - Un dispositivo administrador de energía
    - Un sensor de interfaces analógicas
    - 14 GPIO pins.

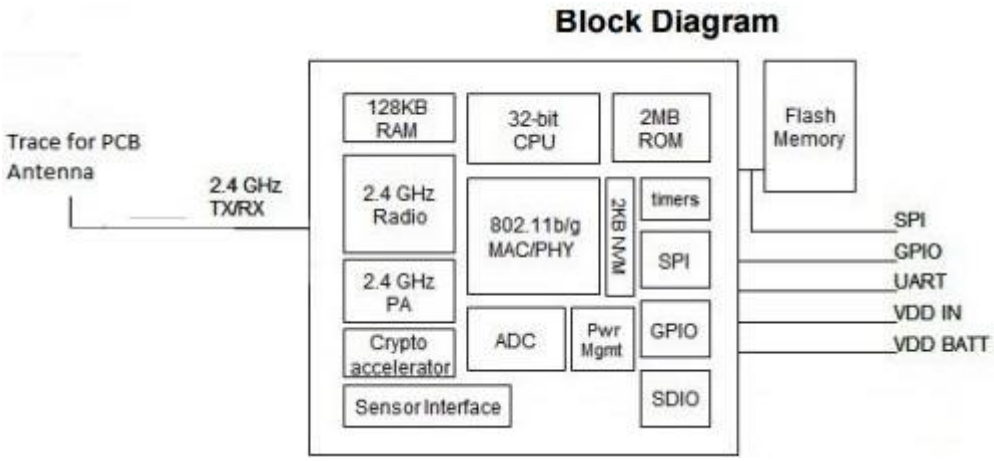

Figura 11.Diagrama del chip WiFly RN171. [19]

3.1.1.6.2 Topologías del módulo WiFi

Las topologías de red que pueden ser desplegadas por medio la API del módulo WiFi son las siguientes:

- **Modo infraestructura**: En este modo los equipos se asocian a la WLAN a través de un punto de acceso y se comunican a través de él. Dentro de las clasificaciones de las WSNs descriptas en el punto 2.1.2.2 podemos clasificar el modo infraestructura como una Flat Network de un único salto.

  Los módulos WiFi son capaces de conectarse contra cualquier Router WiFi estándar configurado en modo Access Point (AP) estándar.

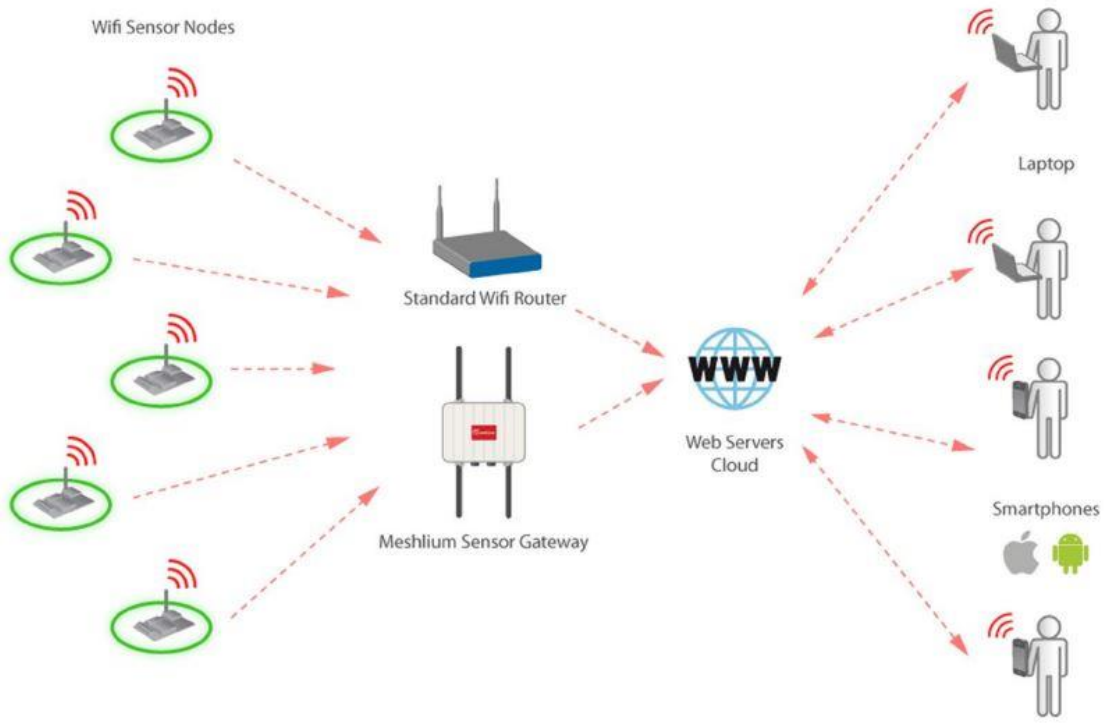

Figura 12. Topología de red módulo WiFi. [14]

- **Modo ad-hoc contra Smartphones**: Este tipo de topología está disponible contra dispositivos Iphone y Android por medio de una APP propietaria. Realmente esta topología es un caso particular del modo infraestructura del punto anterior puesto que el Smartphone debe estar configurado como Access Point. [1]

[1]Este tipo de topología es llamado "modo ad-hoc" en la documentación oficial de Waspmote. Aun así este modo no está relacionado con el modo ad-hoc de la tecnología WiFi.

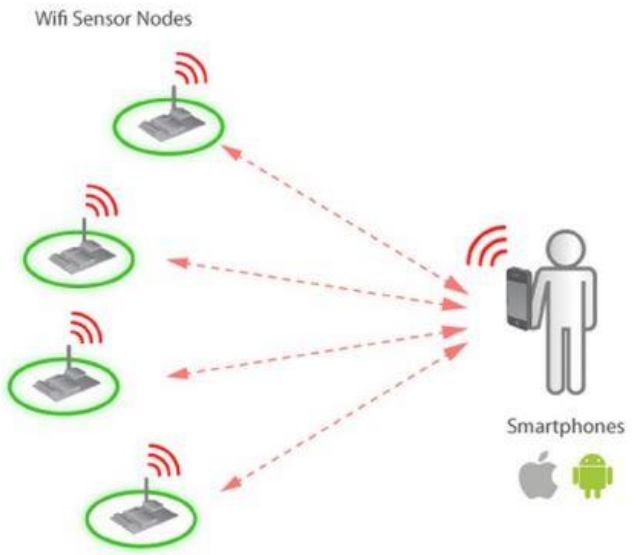

Figura 13. Modo Ad-Hoc contra dispositivos Smartphones. [14]

3.1.1.6.3 Módulo Utilizado

Para la realización del proyecto se ha usado un módulo WiFi junto con la antena 2dBi. A pesar de que el rango de cobertura de la antena de 5 dBi es mayor, la antena de 2 dBi cumple perfectamente con los requisitos necesarios para el proyecto.

3.1.1.7 Módulo RFID

3.1.1.7.1 Funcionamiento de un sistema RFID

El funcionamiento general de un sistema RFID es descrito en el punto 2.2.1.

El módulo RFID del dispositivo Waspmote para poder realizar una operación sobre una tarjeta RFID, deberá encontrarse a una distancia de la tarjeta comprendida entre 5 y 7 cm. En distancias mayores la tarjeta no será reconocida correctamente por el lector RFID del módulo.

El tiempo habitual de ejecutar una operación completa sobre una tarjeta ronda los 80ms. Aun así es recomendable el uso de algún elemento que alerte al usuario del fin de la ejecución de la operación sobre la tarjeta. Una posibilidad es el uso de los LED configurables de la placa para este fin.

3.1.1.7.2 Características RFID

Waspmote dispone de dos módulos RFID diferentes e incompatibles entre ellos, las características de estos módulos pueden resumirse en:

- **RFID/NFC 13.56MHz**. Este módulo tiene las siguientes características:

  - Compatibilidad de lectura y escritura con las siguientes tecnologías ISO 14443A / MIFARE / FeliCaTM / NFCIP-1.
  - Distancia de lectura. 5cm.
  - Capacidad máxima.4KB.

- **RFID 125 KHz.** Este modula tiene las siguientes características.

  - Compatibilidad de lectura y escritura con las siguientes tecnologías ISO cards – T5557 / EM4102. Las tarjetas EM4102 son de solo lectura.
  - Distancia de lectura. 6cm.
  - Capacidad máxima.20B.

3.1.1.7.3 Módulo Utilizado

Para la realización del proyecto hemos utilizado el módulo que funciona a 13.56MHz. La antena de 13.56 MHz dispone de algunas características que hacen su elección más versátil, como:

- La compatibilidad con NFC
- Tarjetas con mayor capacidad de almacenamiento
- Un rango de escritura mayor. Mientras la operación lectura tiene una rango de distancia máxima similar en ambas antenas cuando la operación requiere escribir datos la tarjeta de 125KHz es muy inferior (1cm)
- Mayor velocidad en operaciones de lectura y escritura.

3.1.2 Software

Waspmote, al contrario que otros dispositivos similares, no dispone de sistema operático. Waspmote carga y ejecuta un único programa cada vez. El desarrollo del código de estos programas se realiza empleando el lenguaje C.

Este programa incluye todo el código que Waspmote necesitara en la ejecución, incluyendo, el necesario para acceder a todos los recursos del dispositivo. Durante el desarrollo del programa esta parte del código es incorporada mediante librerías proporcionadas como parte del entorno de desarrollo de Waspmote.

3.1.2.1 Estructura de un programa

Siguiendo las recomendaciones de la documentación oficial de Libelium la estructura de un programa desarrollado para Waspmote consta de 3 partes bien diferenciadas:

- En la primera parte se deben incluir las librerías con las que se va a trabajar, la declaración de las constantes y la declaración de las variables globales.

- En la segunda parte se encuentra la función setup(). Esta función solo es ejecutada una vez durante el tiempo que la placa se encuentra encendida. Siguiendo las recomendaciones esta función se utiliza para el arranque de los diferentes módulos que se van a utilizar en el código.

- En la última parte se encuentra la función loop(). Esta función se ejecuta de manera ininterrumpida en un bucle infinito. El objetivo de esta función es ejecutar de manera secuencial el programa que se ha cargado en la placa.

  Libelium también hace una recomendación a la estructura que debe de tener el código que se ejecuta en la placa dentro de la función loop. En esta recomendación nos encontramos de nuevo 3 partes diferenciadas:

  - La primera parte se reserva para la medida de los datos.

  - La segunda corresponde al envío de los datos.

  - La tercera parte deja la placa en alguno de los modos de bajo consumo (“Sleep”, “Deep Sleep” o “Hibernate”) según se requiera.

```
//1. Incluir librerias
//2. Definición de constantes
//3. Declaración de variables globales

void setup()
{
    //4. Inicialización de módulos
}

void loop()
{
    //5. Realizar medida
    //6. Enviar la información
    //7. Modo Sleep
}
```

Figura 14. Estructura de un programa Waspmote. [20]

3.1.2.2 IDE

Para el desarrollo de los programas que van a ser ejecutados en la placa, Waspmote dispone de un IDE (Integrated Development Environment) que ya incluye la API de la placa. El uso de este software permite:

- Desarrollar, compilar y subir programas a la placa Waspmote sin necesidad de cambiar de entorno de desarrollo.

- Dispone de una interfaz que permite monitorizar los datos enviados por la interfaz serie USB. Esta interfaz supone una gran ayuda en la fase de depuración y búsqueda de errores del código.

- Obtener los ficheros usados para OTA (Over The Air): El IDE genera dos ficheros en la compilación del programa un fichero hexadecimal y uno binario:

    - El fichero hexadecimal: puede ser utilizado para el OTA de redes con nodos equipados con módulos ZigBee. En este caso, el software es enviado a la placa y almacenado en la tarjeta SD por el nodo. Tras esto el nodo recibirá un comando de "start_new_program", para a continuación cargar el programa de la tarjeta SD a la memoria Flash desde donde se ejecutara el nuevo programa.

    - El fichero binario puede ser utilizado para el OTA de redes con nodos equipados con módulos 3G/GPRS/WiFi. En este caso, es la placa la que

debe iniciar la comunicación para obtener el binario del programa, realizando consultas a un servidor FTP.

- Evita sobrescribir el programa de arranque que se encuentra en la memoria flash a la hora de subir el programa.

3.1.2.3 Librería WiFi

En este punto queremos exponer un pequeño análisis de la librería WiFi incluida en el API de la plataforma. La razones son primero conocer su posibilidades de uso y segundo dejar constancia que el chip rn171 del módulo WiFi implementa posibilidades de configuración que la librería todavía no utiliza en su código.

La estructura de la librería consta de una serie de funciones que permiten ejecutar los comandos de configuración del chip rn171. Los comandos implementados en la librería se encuentran almacenados en el array "table_WIFI". Podemos agrupar las funciones en:

- **Funciones set:** Como "setESSID", "setGW", "setNetmask", etc. Estas funciones siguen la siguiente estructura:

  1. Se genera la cadena de envío que contiene el comando del chip. Para ello se utilizan los métodos "strcpy_P" y "snprintf".
  2. Se envía la cadena haciendo uso de la función "sendCommand". La función sendCommand se encarga de reiniciar la comunicación serie sobre la UART, enviar el comando y esperar la respuesta.

  Esta estructura sufre muy poca variación en las funciones set. En ocasiones consiste en repeticiones de la misma estructura si la función requiere varios comandos para llevar a cabo la configuración. Además, esta estructura suele ser parte de la estructura del resto de funciones que ejecutan comandos en el chip.

```
//! Sets the netmask.
uint8_t WaspWIFI::setNetmask(ipAddr netmask)
{
    char question[128];
    char buffer[20];

//Punto 1º  ////////////////////////////////////////////////////////////
    // Copy "set i n "
    strcpy_P(buffer, (char*)pgm_read_word(&(table_WIFI[31])));
    snprintf(question, sizeof(question), "%s%s\r", buffer, netmask);
//Punto 2º  ////////////////////////////////////////////////////////////
    return sendCommand(question);
}
```

Figura 15. Estructura típica de las funciones "set" de la librería.

- **Funciones get:** Como "getURL", "getRSSI", "getStats", etc. La estructura de estas funciones se puede dividir en:

1. Generación de la cadena de envío que contiene el comando y envío de la cadena sin reiniciar la comunicación con la UART.
2. Esperar la respuesta de la UART.
3. Almacenar la información recibida en la variable "answer".

```
//! Displays current network status, association, authentication, etc.
void WaspWIFI::getAPstatus()
{
    char question[20];
    uint16_t i=0;

 //Punto 1º ////////////////////////////////////////////////////////////////////
    // Copy "show n\r"
    strcpy_P(question, (char*)pgm_read_word(&(table_WIFI[75])));

    // Sends the command.
    serialFlush(_uartWIFI);

    printString(question,_uartWIFI);
//Punto 2º ////////////////////////////////////////////////////////////////////
    // Waits an answer.
    i=strlen(question);
    i=i+5;
    if(!waitForData(i))
    {
        return (void)0;
    }
    i=0;
//Punto 3º ////////////////////////////////////////////////////////////////////
    while(serialAvailable(_uartWIFI))
    {
        if (((answer[i]=serialRead(_uartWIFI))!='\0')&&(i<(sizeof(answer)-1)))
        {
            i++;
        }
        delay(10);
    }
    answer[i]='\0';
}
```

Figura 16.Estructura típica de las funciones "get" de la librería.

- **Resto Funciones:** El resto de funciones no siguen una estructura definida. Aunque, cuando deban de ejecutar algún comando sobre el chip rn171, seguirán alguno de los métodos descritos.

Como se ha comentado antes el chip rn171 implementa en su firmware algunas posibilidades que están sin explotar todavía por la plataforma. Una de estas posibilidades disponibles en el chip y no implementadas todavía en la librería es la de permitir configurar el chip en un modo punto de acceso[2]. La implementación de este modo permitiría, por ejemplo, el despliegue de redes jerárquicas de nodos Waspmote con el módulo WiFi. Este hecho tendrá cierto impacto en la parte de análisis y diseño de la solución del sistema de control de inventario.

[2] Disponible a partir de la versión 2.45 del firmware del chip rn171, implementado en sustitución del modo ad-hoc del que disponía el chip. El modo ad-hoc solo está disponible en versiones anteriores a la versión 2.36 del firmware.

3.2 Interfaz Web

En la implementación del sistema de gestión de inventario, con la intención de facilitar el acceso al servidor, se ha optado por utilizar una solución web como interfaz del sistema de "back-end". Por todo esto, en este punto se analizan las tecnologías más relevantes utilizadas en el desarrollo de la interfaz web.

3.2.1 HTML y CSS

HMTL (Hyper Text Markup Language) y CSS (Cascading Style Sheets) definen el contenido y la apariencia de una página web respectivamente. A lo largo de los años la evolución de estas dos tecnologías se ha producido de manera paralela y en las versiones aquí analizadas, las más recientes, es donde su relación se ha visto fortalecida más que nunca.

3.2.1.1 HTML5

Realmente un documento HTML es simplemente texto con marcas. Estas marcas, también llamadas etiquetas, permiten realizar acciones específicas sobre el texto como, por ejemplo, generar enlaces o incluir imágenes. Históricamente el desarrollo HTML ha dado muchas vueltas y durante algún tiempo parecía que XHTML (una versión más formal de HTML) iba a ser el camino a seguir en la evolución del estándar web. Pero en 2004 con la formación del WHATWG (Web Hypertext Application Technology Working Group) un grupo de trabajo formado por empresas de la industria y en 2006 cuando se anunció la creación de un nuevo grupo de trabajo en el W3C (World Wide Web Consortium) HTML continuo su desarrollo con HTML5. Finalmente en octubre de 2014 el W3C publica el documento "HTML5: A vocabulary and associated APIs for HTML and XHTML" [21] que da forma a la última versión del estándar de hipertexto para web, HTML5.

A continuación se describen algunas de las novedades más importantes de HTML5 respecto a versiones anteriores:

- **Multimedia:** HTML5 introduce la posibilidad de integrar contendido multimedia mediante la inclusión de etiquetas que permite la incrustación de contenido multimedia directamente en el código HTML.:
  - La etiqueta <audio> permite incluir archivos de audio con formato MP3, Wav y Ogg.
  - La etiqueta <video> permite incluir ficheros de video con formato MP4, WebM y Ogg.
  - La etiqueta <canvas> permite incluir dibujos generados dinámicamente por CSS o por Javascript.

- **Cambios:** Se han realizado múltiples cambios y eliminado en un gran número de etiquetas de versiones anteriores. Algunas de las más representativas:

    - **input**: Tal vez, el cambio más representativo sobre la etiqueta "<input>" sea el gran número de valores nuevos añadidos al atributo type que simplifican y en algunos casos sustituyen el posterior chequeo de formularios con Javascript, a la que se han añadido una gran cantidad de nuevos atributos. Ahora el atributo puede tomar, por ejemplo, valores como number, en el que el input solo puede almacenar un número o email, que valida el formato de la entrada esperando un email.

    - **CSS**: Como hemos comentado anteriormente estas dos tecnologías han crecido en paralelo y su relación en estas versiones se hace más estrecha. Hasta el punto de que se han eliminado etiquetas en favor de tareas que CSS es capaz de realizar. Algunas etiquetas eliminadas en favor del uso de CSS son, "<basefont>", "<big>", " ", "<font>", "<strike>" o "<tt>". CSS será explicado después en el punto 3.2.1.2.

- **APIs:** La inclusión de APIs que permiten la inclusión de funcionalidades adicionales como realizar Drag&Drop [22], geolocalización o ejecuciones en segundo plano con Web Workers [23]. Estas funcionalidades requieres del uso de Javascript para llevarse a cabo.

- **Web semántica:** HTML5 incluye etiquetas nuevas para el manejo de la Web semántica (Web3.0). La Web semántica hace referencia al significado del contenido, su finalidad o sus relaciones. Las etiquetas nuevas son "<article>", "<aside>", "<details>", "<figcaption>", "<figure>", "<footer>", "<header>", "<main>", "<mark>", "<nav>", "<section>", "<summary>" y "<time>".

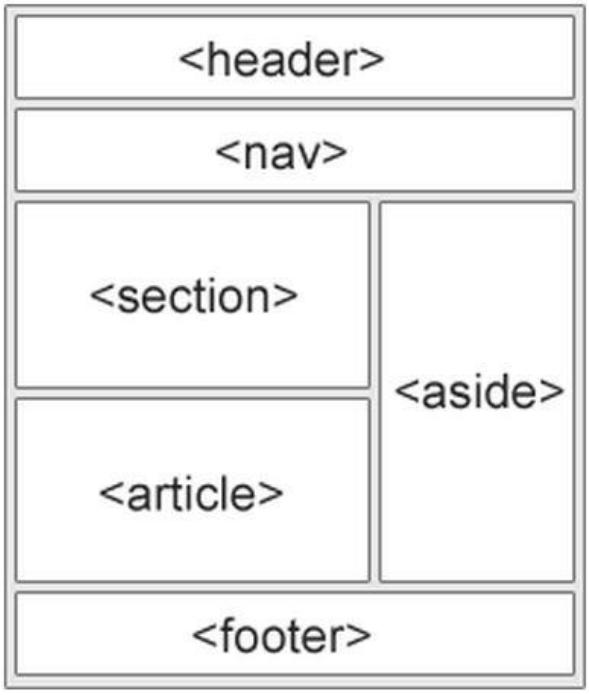

Figura 17. Diferentes partes de una página web con elementos semánticos. [24]

3.2.1.2 CSS3

CSS es un lenguaje creado para generar la presentación visual de un documento HTML y XML. CSS viene a sustituir los atributos referentes a la presentación visual que eran utilizados en versiones antiguas de HTML y que dificultaban la accesibilidad de la página. Además CSS permite mantener un control centralizado sobre la presentación de la web lo que facilita el desarrollo y los posibles cambios de presentación futuros.

Aunque la recomendación de CSS3, también llamado CSS nivel 3, no ha sido todavía finalizada definitivamente, todos los meses se publican borradores de las futuras recomendaciones del estándar. Esto es debido a que al contrario de las versiones CSS1 y CSS2.1, la recomendación de CSS3 se han dividido en módulos (más de 30). Cada módulo añade nuevas funcionalidades o extiende las funcionalidades de versiones anteriores.

3.2.2 JavaScript y AJAX

3.2.2.1 JavaScript

JavaScript es un lenguaje de script originalmente creado para Netscape. JavaScript toma su nombre del hecho de poseer una sintaxis similar al lenguaje de alto nivel Java. Posteriormente Netscape delegó en la ECMA (European Computer Manufacturers Association) la normalización del lenguaje. Actualmente el nombre real del lenguaje es ECMAScript, aunque todavía suele ser llamado JavaScript a pesar de ello. De hecho JavaScript es la adaptación soportada por Mozilla de ECMAScript, al igual que jscript es la adaptación soportada por Microsoft.

JavaScript es ejecutado en el lado del cliente para realizar operaciones sobre la página web ya descargada. Con JavaScript se puede modificar el contenido de una etiqueta, cambiar sus atributos, modificar el estilo de la página. Actualmente su combinación con AJAX (Asynchronous JavaScript and XML) permite el envío y recepción de datos de manera dinámica.

3.2.2.2 AJAX

AJAX es una técnica de desarrollo de páginas web para crear páginas web rápidas y dinámicas. AJAX brinda la posibilidad de hacer consultas, inserciones, modificaciones sobre datos del servidor sin la necesidad de recargar de manera completa la página actual sobre la que el usuario esta interactuando. Para llevar esto acabo, en segundo plano, AJAX permite mantener una comunicación asíncrona con el servidor web para el intercambio de datos.

AJAX utiliza el objeto XMLHttpRequest para poder realizar la petición de los datos al servidor de la página web. Actualmente el objeto XMLHttpRequest es soportado por todos los navegadores modernos.

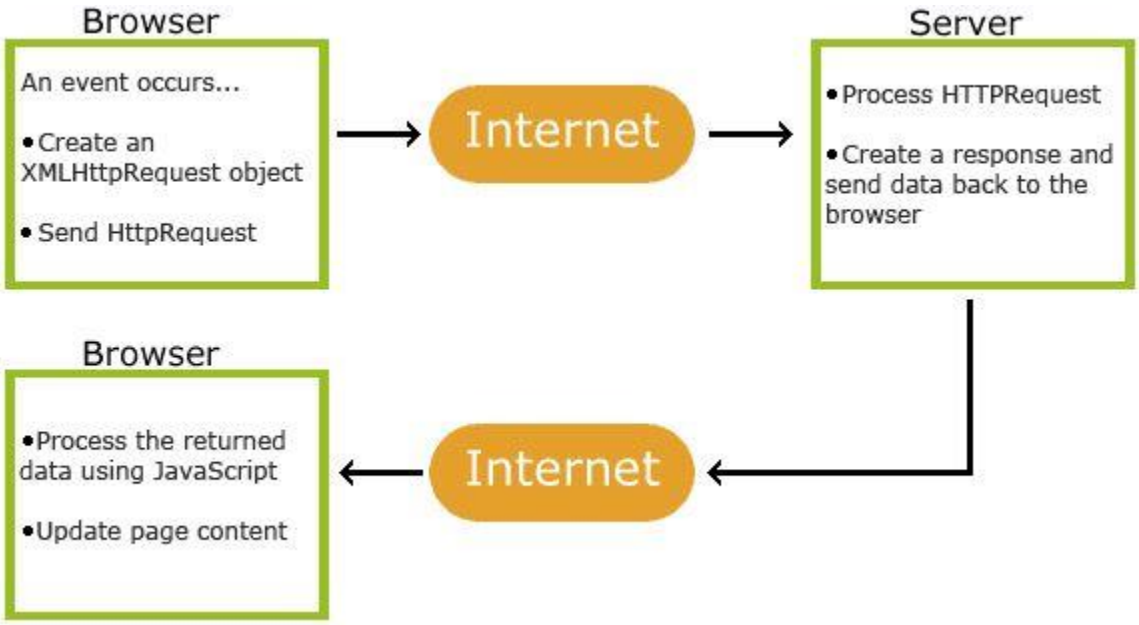

Figura 18. Ciclo del funcionamiento de una consulta AJAX. [25]

3.2.3 Java Beans, Servlets y JSPs

3.2.3.1 Java Beans

Según la documentación oficial de Sun Microsystem, "Un Java Bean es un componente de software reutilizable que puede ser manipulado visualmente en una herramienta contenedora" [26].

Los Java Beans son clase de java usadas para modelar datos abstractos como por ejemplo, un objeto Usuario compuesto de varios campos como su nombre, su apellido, su login, etc. Existe una convención que define las características que debe cumplir un Java Bean:

- Poseer un constructor sin argumentos.
- Los atributos del Java Bean deben de ser privados.
- Debe disponer de métodos get y set para poder acceder a los atributos de clase. Estos métodos siguen una convención en su nomenclatura en que tanto el método get como set deben ir seguidos del nombre del atributo al que aplican, por ejemplo para el caso del objeto Usuario el atributo nombre deberá disponer de un método getnombre y otro setnombre.
- Deben implementar la clase java.io.Serializable.

3.2.3.2 Servlets

Los Servlets tienen entre sus usos más comunes el desarrollo de aplicaciones web generadas en el lado del servidor. Un servlet se implementa como una clase que extiende de la clase HttpServlet. Es necesario que contenga un método que sobrescriba al menos uno de los métodos de la clase HttpSerlvet. Normalmente el implementado es uno de los siguientes métodos doGet, doPost, doPut, doDelete, init, destroy y getServletInfo. Alguna de las ventajas de los servlets respecto otras tecnologías, son:

- Optimización de recursos:
    - Los servlets son instanciados una única vez y se encuentran cargados en memoria permanentemente.
    - Las distintas peticiones son ejecutadas en hilos diferentes en lugar de en procesos distintos.
- Corren sobre una Java Virtual Machine (JVM): Esto incluye todas las ventajas inherentes a Java, portabilidad, gestión de memoria, API bien definida y documentada.

3.2.3.3 JSP

Una Java Server Page (JSP) es una tecnología para el desarrollo de páginas web dinámicas que en su código incluye fragmentos de un lenguaje de marcado como HTML, XML, XHTML y otras partes del mismo es Java. La primera vez que una JSP es invocada es traducida al código de un Servlet, por tanto, en las siguientes ejecuciones de dicho JSP será el Servlet nuevo el que se ejecuta produciendo como salida el código HTML que compone la página web de la respuesta. Se puede decir entonces, que las ventajas aplicadas a Servlets también se aplican a las JSPs.

Estas tres tecnologías suelen trabajar juntas siguiendo el patrón de arquitectura de software MVC (modelo-vista-controlador) donde cada una de ellas asume un papel del patrón:

- Los Java Beans son el Modelo de la arquitectura MVC, son los objetos que representan la información con la que el sistema trabaja.
- Las JSP son la Vista de la arquitectura MVC, es la entidad que termina presentando el modelo, en la mayoría de ocasiones actuando como interfaz de usuario. Puesto que, como hemos dicho, se encarga de mostrar los datos modelados debe de tener acceso a los mismos, por tanto debe de tener acceso a los Java Beans.
- Los Servlets son las clases que representan el controlador en el MVC. Es la entidad que responde a eventos, generalmente acciones de un usuario, y actúa

generando instancias sobre el modelo o incluso actuando sobre la propia vista si se solicita un cambio en la misma. Los servlets son el intermediario entre los JSP (vista) y los Java Beans (Modelo)

3.2.4 Apache Tomcat

Apache Tomcat es un servidor web y un contenedor de Servlets de código abierto. Apache Tomcat es una implementación de las tecnologías Java Servlet, Java Server Pages (JSP), Java Expression Language and Java WebSocket.

Tomcat es apto para su uso como servidor Web autónomo en entornos empresariales con gran nivel de tráfico y entornos de alta disponibilidad. A pesar de ello, también puede funcionar junto con otros servidores Web externos como Apache, Microsoft Internet Information Server (IIS), Netscape Enterprise Server.

Se consideraron otras opciones, como JBoss, para albergar los Servlets y JSPs. Pero finalmente se seleccionó Tomcat puesto a que es una tecnología con la que ya teníamos cierta experiencia previa y era perfectamente funcional para los objetivos del proyecto. Para el desarrollo del presente proyecto, Tomcat es utilizado como servidor web y como contenedor de Servlets. La versión de Tomcat instalada en el entorno de desarrollo del proyecto es la versión 7 que implementa la versión de Servlets 3.0.

3.2.5 Mysql

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. MySQL es una de las bases de datos más populares para su uso en aplicaciones Web, se caracteriza por un gran rendimiento, fiabilidad de uso y sencillez. MySQL implementa el lenguaje estándar para realizar consultas a bases de datos relacionales SQL (Structure Query Language).

MySQL es una base de datos de código abierto que se ofrece bajo la licencia GNU GPL para usos compatibles con la misma. Para usos en el entorno empresarial requiere adquirir licencia específica.

3.3 Conclusiones del capítulo

En este capítulo se han descrito las herramientas y tecnologías más importantes utilizadas en el proyecto. Listamos, de nuevo, las tecnologías seleccionadas para el proyecto:

- Waspmote: Es el dispositivo a estudio. Además, también son los nodos de nuestra red. En su configuración hardware disponen de:
    - Módulo WiFi: Es el módulo de comunicaciones escogido para el desarrollo.
    - Módulo RFID: Los nodos de nuestra red carecen de elementos sensores. RFID es la encargada de obtener los datos a reportar a nuestro servidor del sistema.
- HMTL5 y CSS3: Son los lenguajes utilizados en la parte Web de nuestro sistema.
- JavaScript, AJAX: Permitir añadir cierto dinamismo a los elementos de nuestra aplicación web en el lado del cliente.
- Java Beans, Servlets y JSPs: Conforman las tecnologías que corren en el lado del servidor en nuestro sistema de control de inventario.
- Apache Tomcat: Es el servidor web que alberga la aplicación desarrollada.
- Mysql: Es la base de datos del sistema.

Las tecnologías aquí descritas junto con algunas otras han sido necesarias para el diseño y desarrollo de nuestro sistema de inventario. En los capítulos “Análisis y Diseño” y “Desarrollo” se describirá el papel y uso de cada una de ellas en la implementación final del sistema de control de inventario.

Capítulo 4

Análisis y Diseño

Este capítulo aborda el análisis llevado a cabo previamente al diseño del sistema. Y pretende describir y justificar todas las decisiones tomadas a la hora de llevar a cabo el diseño del sistema de control.

4.1 Análisis de la solución

En el análisis previo al diseño de la solución es necesario considerar lo comentado con anterioridad en los objetivos del proyecto. Concretamente en el punto 1.2, en donde se comenta que la naturaleza inicial del proyecto tenía dos objetivos principales. El primero era obtener experiencia sobre las capacidades del hardware de Waspmote y su funcionamiento conjunto con el chip WIFI WiFly rn171 y la antena RFID. Y la segunda era utilizar lo aprendido en la primera fase para llevar a cabo un desarrollo de un sistema de control de inventario.

Como también se señala en los objetivos del proyecto, en ningún momento se ha pretendido desarrollar una herramienta de control de inventario completa sino ver las posibilidades de Waspmote en una aproximación a un sistema de control de inventario.

Aun así, el análisis del sistema ha pretendido llevar cierto orden en su desarrollo como se muestran en los puntos a continuación. Y se ha puesto esfuerzo en realizar un análisis de requisitos previos al diseño del sistema.

4.1.1 Topología de la solución

El primer paso en el análisis del sistema del control de inventario ha sido decidir la topología en el que se han desplegado los nodos Waspmote. Se han barajado dos posibles diseños para incluir Waspmote en la solución de control de inventario:

1. **Red Jerárquica:** En esta solución se opta por definir una red jerárquica en la que cada zona a inventariar corresponde con uno o varios Cluster Head. El nodo o nodos Cluster Head de una zona concreta serán los encargados de procesar los datos y hacérselos llegar al servidor identificando la zona desde la que llega el paquete de datos.

   La Figura 19 muestra un ejemplo de funcionalidad de esta solución. El área de cobertura de los "Waspmote de zona" define la extensión de una zona. Cuando un "Waspmote lector" se desplaza realizando lecturas es el "Waspmote de zona" el que recibe la lectura y es el encargado de enviarlo al servidor a través de la red inalámbrica.

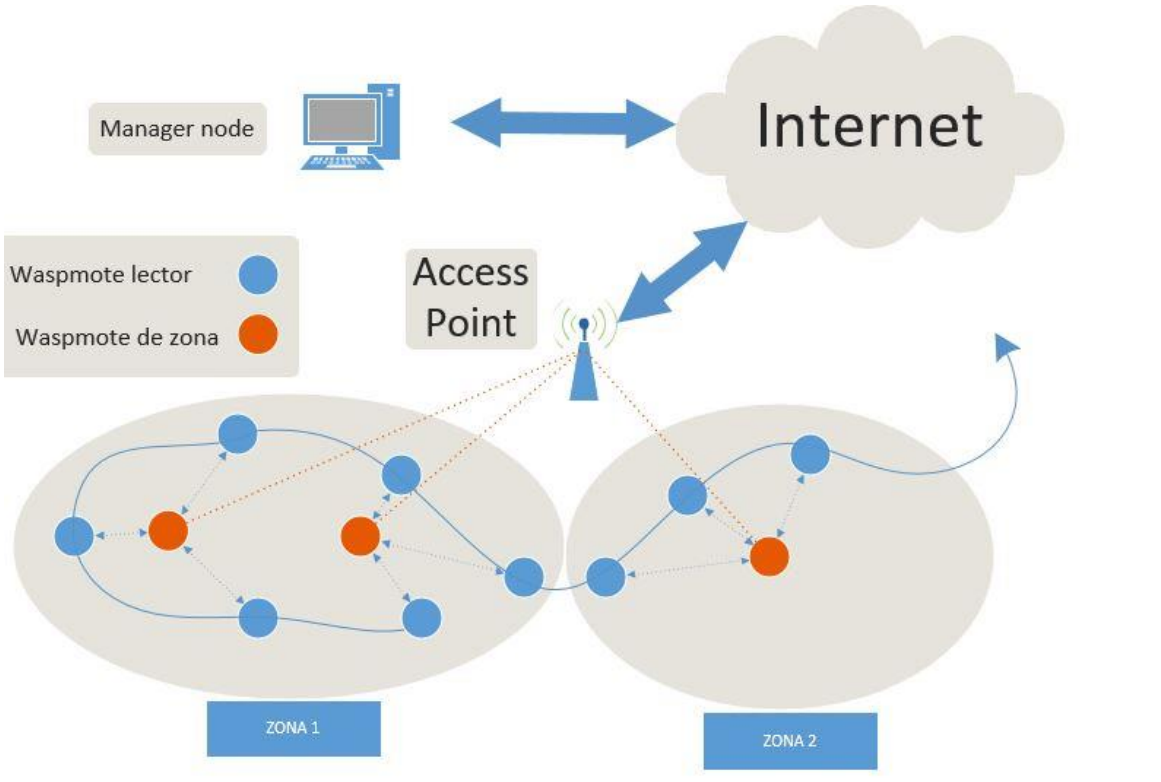

Figura 19. Esquema de la solución de red jerárquica

2. **Red plana:** Una red homogénea donde todos los nodos realizan la misma función. Las zonas se identifican con etiquetas RFID al igual que los ítems a inventariar, para inventariar los ítems de una zona previamente debe leerse la RFID de la zona que se está leyendo. La topología WiFi utilizada para esta solución sería la de modo-infraestructura que permite la librería de la API de Waspmote.

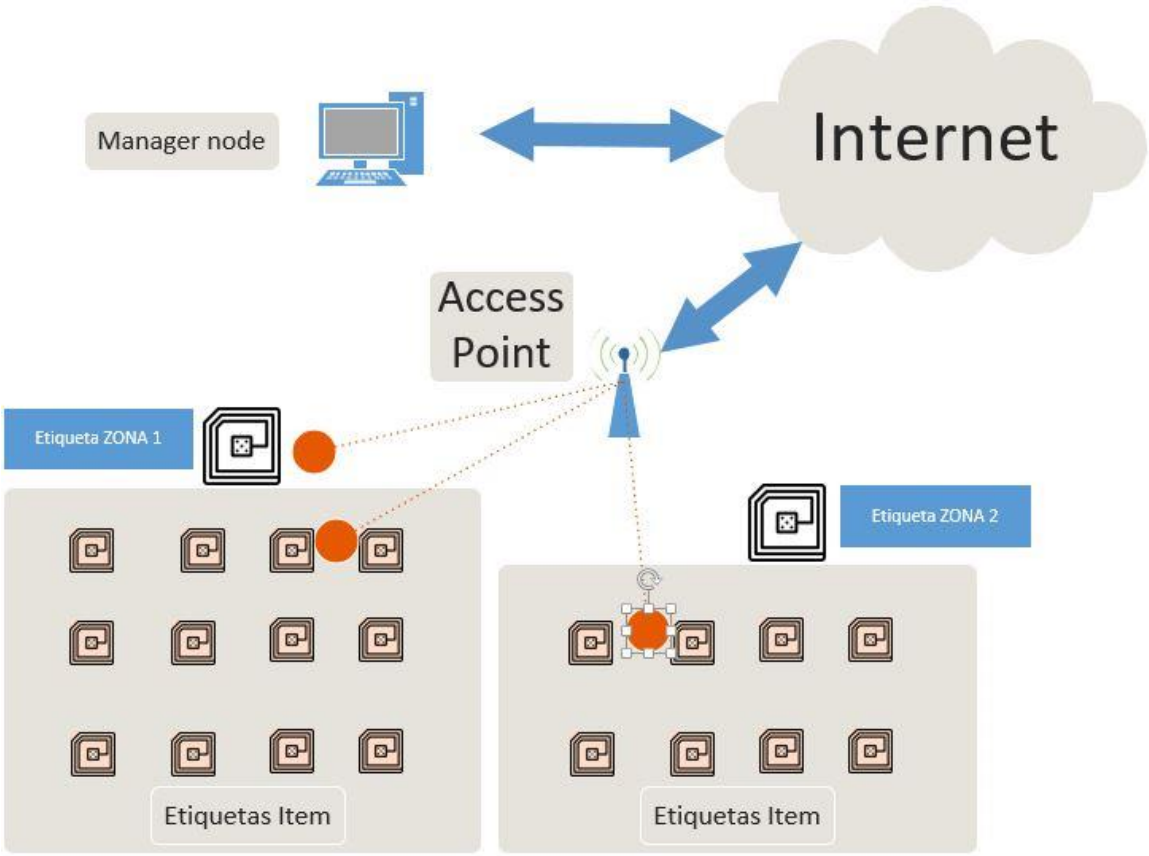

Figura 20. Esquema de la solución de Red Plana

En esta solución el inventario requiere que antes de inventariar los ítems de una zona, se lea la etiqueta referente a la zona. A partir de ahí todos los ítems leídos a continuación se identificaran como asociados a esa zona. La Figura 20 intenta describir lo expuesto en el anterior párrafo.

Tras exponer ambas posibilidades a debate, finalmente, es la segunda opción la que decide llevarse a cabo. Las razones que nos han hecho acabar decantándonos por esta decisión son:

- Costes: La opción número 1 exigiría la necesidad de incluir nodos Waspmote actuando de Cluster Head para identificar zonas. Esto supone un coste significativamente mayor que hacerlo mediante una tarjeta RFID.

- Sencillez: El despliegue, el desarrollo, y la administración del sistema se ven significativamente reducidos:

    - Despliegue: Incluir una nueva zona simplemente implica añadir una nueva etiqueta al sistema.

    - Desarrollo. La opción 1 requiere la necesidad de hacer desarrollo con la placa Waspmote al margen de la librería WiFi de la API. Algunos de estos desarrollos abordan desafíos que hemos decidido dejar fuera del alcance del proyecto, por ejemplo:

- En la opción 1 existe la necesidad de poder configurar el módulo WiFi de los nodos "Waspmote de zona" en modo access point. Como se indica en el punto 3.1.2.3 la librería WiFi no implementa la posibilidad de configurar el chip rn171 en modo AP. Pero el chip sí que dispone de esa posibilidad.

  Por lo que, como se ha comentado, la implementación de la red jerárquica de la opción número 1 requeriría desarrollo al margen de la librería de la plataforma.

- La opción 1 también plantea desafíos como es la dificultad de limitar las zonas con la cobertura de los nodos Cluster Head o implementar algoritmos de enrutado más complejos.

4.1.2 Marco de la solución

En este punto se indica cuál será el campo de actividad que pretende cubrir el sistema de control de inventario.

El diseño de la solución del proyecto se orienta a la realización de inventarios en un entorno con equipos ofimáticos, como una universidad o una oficina. Aunque la aplicación podría ser adaptada a otros entornos tras la adaptación de algunos formularios web y los pertinentes cambios en la base de datos.

La Figura 21 muestra un ejemplo del entorno objetivo de funcionamiento del sistema de control de inventario.

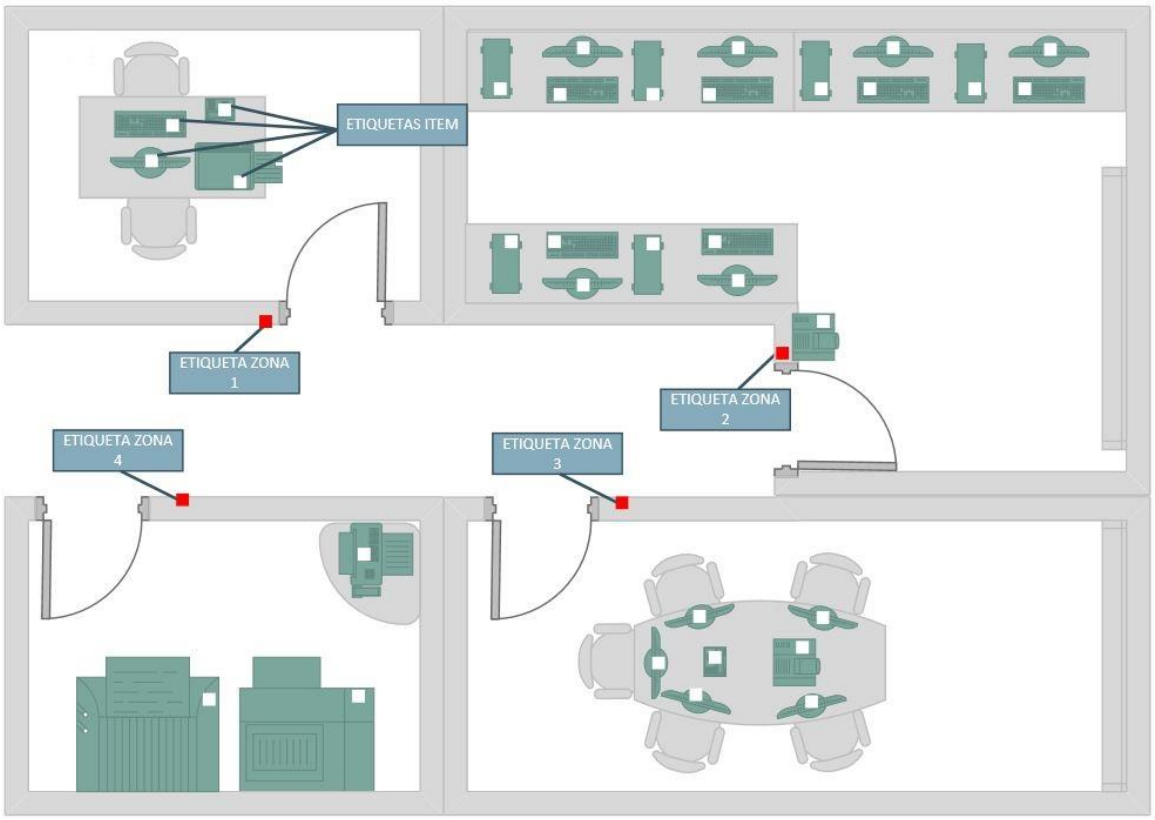

Figura 21. Ejemplo de escenario del proceso de inventario

4.1.3 Casos de uso de la aplicación Web

En este apartado se expondrán los casos de uso de los que se ha partido para el desarrollo del sistema. Los casos de uso permiten identificar la interacción de los usuarios con el sistema y ayudan a definir los requisitos funcionales del sistema.

Los actores que se distinguen en los casos de uso son los distintos roles de los que disponen los administradores de la aplicación web.

4.1.3.1 Casos de uso de la pantalla de login

En la Figura 22 se muestra el caso de uso que se contemplan en la pantalla correspondiente a la interfaz de login del sistema.

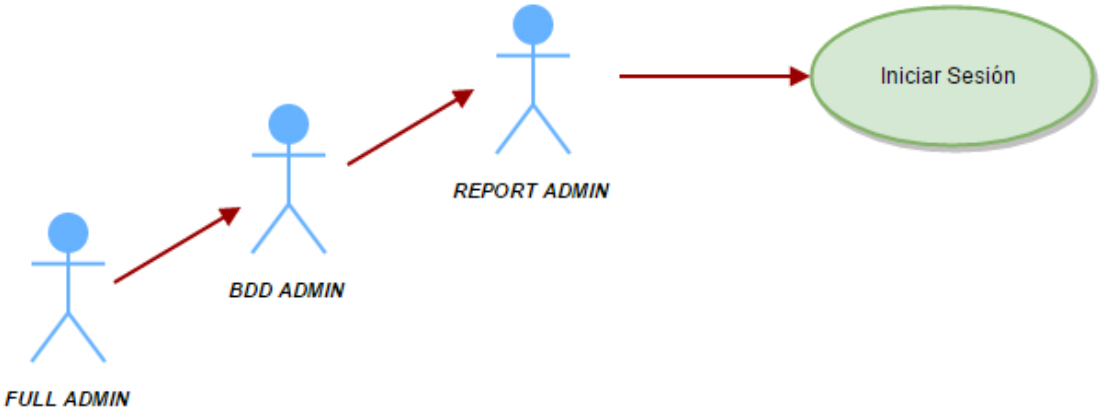

Figura 22. Caso de uso de la pantalla de login

4.1.3.2 Casos de uso de la pantalla principal

En la Figura 23 se muestra el caso de uso que se contempla en la pantalla correspondiente principal de la web de administración del sistema. En la figura se contemplan las distintas posibilidades de las que disponen los distintos administradores.

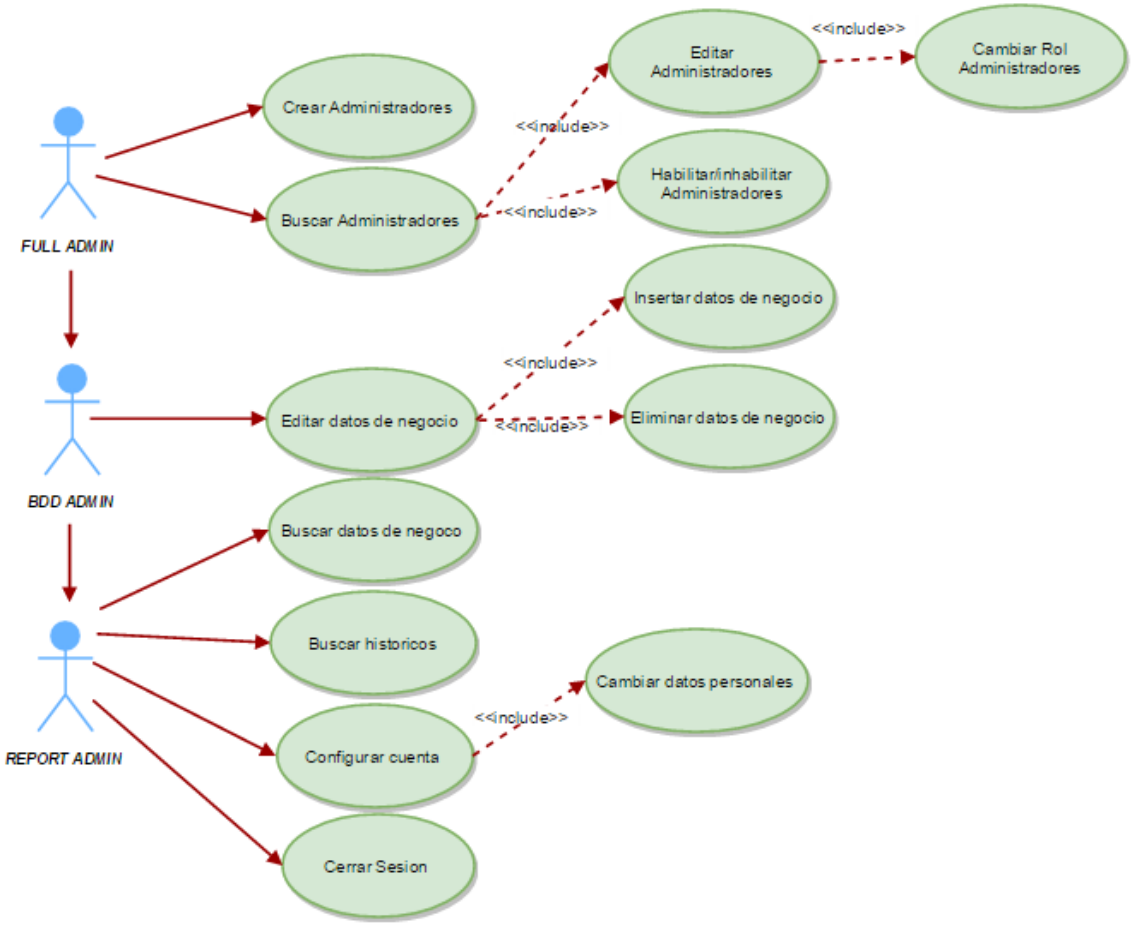

Figura 23. Caso de Uso de la ventana principal.

4.1.4 Requisitos del sistema de control de inventario

Para terminar con la fase de análisis y empezar a plantear la fase de diseño se hace necesario definir unos requisitos previos de los que poder partir. En este punto se muestra una especificación no formal de requisitos obtenidos tras diversas reuniones con el tutor del proyecto.

Es conveniente aclarar que, aunque los requisitos aquí listados hacen referencia al sistema completo, los requisitos de la placa Waspmote quedan fuera del alcance de este punto. Si bien es cierto que muchos de las características aquí descritas hacen referencia y afectan directamente a la funcionalidad del programa de la placa Waspmote por su interoperabilidad con el servidor del entorno, los requisitos de la placa Waspmote se describirán en el punto 4.1.5.

Como ya se ha dicho, los requisitos aquí descritos siguen un formato no formal y son mostrados como una lista para facilitar la lectura. La lectura detallada de los requisitos se encuentra en el Anexo I. Requisitos.

Los requisitos iniciales son:

- Implementar una consola de administración web, que permita administrar los datos de negocio de la base de datos. Realizar búsquedas, insertar, actualizar y eliminar datos.

- Implementar un sistema para la gestión de usuarios administradores. Que permita crear, eliminar, modificar y desactivar usuarios.

- Implementar un conjunto de roles de administración que concedan diferentes niveles de acceso a la aplicación.

- La aplicación almacenar los históricos de los inventarios realizados. Permitirá consultarlos, pero no permitirá modificarlos ni eliminarlos.

- El sistema contara con un método que permita recibir los envíos de los dispositivos Waspmote utilizando el protocolo HTTP.

- El sistema permitirá desactivar la recepción de mensajes de un Waspmote concreto.

- El sistema permitirá a los dispositivos Waspmote sincronizarse con el servidor utilizando el protocolo NTP.

- El sistema deberá poder enviar correos de alerta a los responsables de un ítem cuando se produzca un cambio de ubicación del mismo.

- Se implementara algún tipo de seguridad en la encriptación de las contraseñas de los usuarios administradores.

- La aplicación deberá ser desarrollada utilizando las tecnologías de Servlets, JSPs, AJAX, JavaScript, HTML5 y CSS3.

- El entorno de desarrollo será Ubuntu 12.04LTS.

- El desarrollo del proyecto se llevara a cabo utilizando Software libre siempre que sea posible.

4.1.5 Requisitos de la placa Waspmote

Para el diseño del software de la placa Waspmote no se ha partido de unos requisitos formales puesto que estos, así como el análisis y diseño de la aplicación que se ejecuta en el dispositivo Waspmote, se han originado a medida que se adquiría conocimientos con la tecnología.

Finalmente se parte de los siguientes requisitos:

- **Sincronización NTP:** Los nodos deben de sincronizarse con el servidor. Si una placa no se encuentra sincronizada no podrá enviar las lecturas RFID al servidor.

- **HTTP:** El nodo deberá de enviar al servidor las lecturas realizadas utilizando el protocolo HTTP. La request HTTP deberá llevar una marca de tiempo.

- **SD:** Las lecturas que no se hayan enviado correctamente, bien por una respuesta negativa del servidor o por una respuesta diferente de la 200 OK, serán almacenadas en la tarjeta SD.

4.2 Diseño de la solución

En este punto se muestra el diseño final llevado a cabo en el desarrollo del proyecto a partir de la fase de análisis. Teniendo siempre presente lo expuesto en los requisitos de usuario y los casos de uso y los objetivos del proyecto.

4.2.1 Base de datos

Antes de realizar el diseño de la base de datos se hace necesario organizar la información que va a ser necesario almacenar en la base de datos. Los datos a almacenar deben de cumplir las siguientes condiciones:

- Los objetos a inventariar se definen como “Ítems”.

- Las áreas a inventariar se definen como “Zonas”.

- Los “Ítems” se encuentran en “Zonas”.

- Existen etiquetas RFID y todas ellas se identifican por su “UID”.

- Una etiqueta identifica a un “Ítem” o a una “Zona”.

- Las etiquetas que identifican un “Ítem” guardan la “Zona” en la que se encuentran actualmente y el “Ítem” al que representan.

- Las etiquetas que identifican una “Zona” guardan la “Zona” a la que actualmente representan.

- Todo “Ítem” pertenece a un responsable.

- Toda “Zona” tiene uno o varios responsables.
- Los dispositivos “Waspmote” se identifican por su mac y por el identificador del dispositivo.
- Los históricos almacenan la información del dispositivo “Waspmote” que envío los datos y la “Zona”.
- A los administradores de la aplicación web se les llama “Manager”.
- En ocasiones los Manager podrán activar sus cuentas mediante una URL enviada a su dirección de correo.

Vista la naturaleza de los datos parece que es posible separar los datos de la lógica de negocio de los datos de administración de la aplicación web. Por lo que, a la hora de diseñar los diagramas entidad-relación se ha decidido separar la información relativa a la solución de la aplicación de inventario y la información relativa a los usuarios administradores en diagramas diferentes. Más adelante esto se traducirá en bases de datos distintas.

La Figura 24 muestra el diagrama entidad-relación de la base de datos de la lógica de negocio.

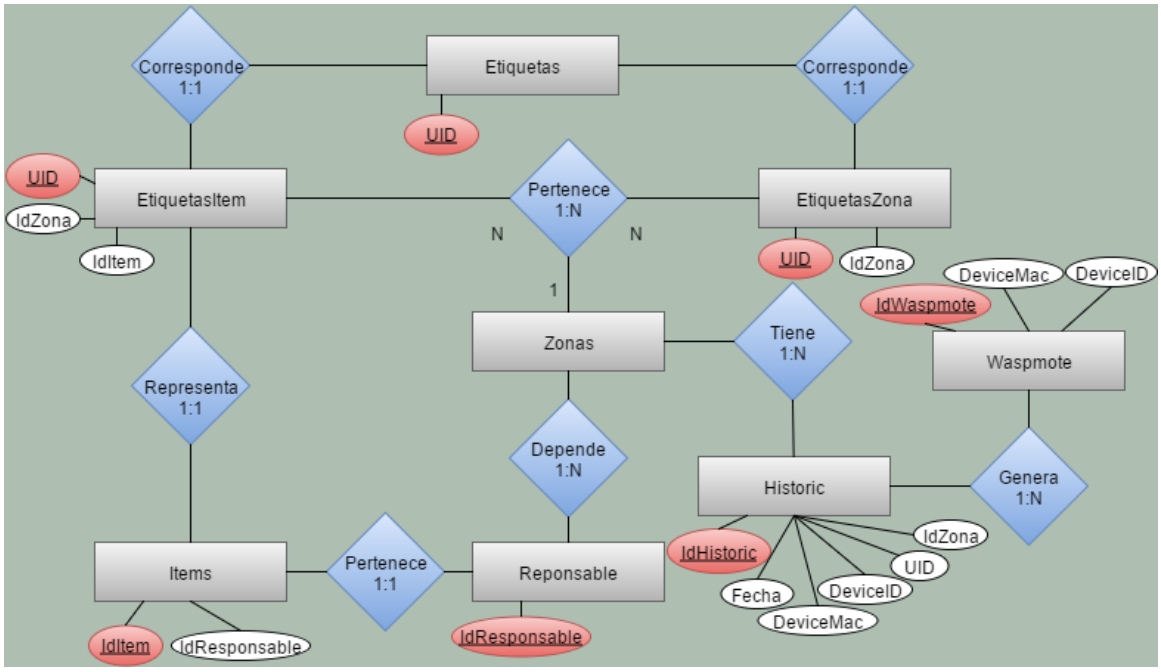

Figura 24. Diagrama entidad-relación de la base de datos de la aplicación.

La Figura 25 muestra el diagrama entidad relación de la base de datos de administradores de la interfaz web.

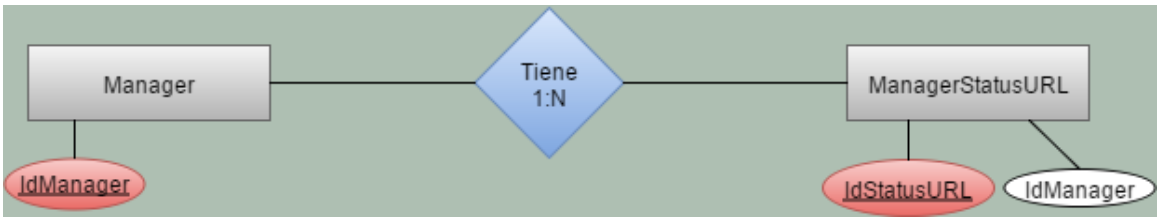

Figura 25. Diagrama entidad-relación de la base de datos de administrador web.

4.2.2 Waspmote

Para el diseño del programa de las placas Waspmote se ha partido de los requisitos descritos en el punto 4.1.5.

La Figura 26 contiene el diagrama de flujo en pseudocódigo previo a la implementación del diseño final.

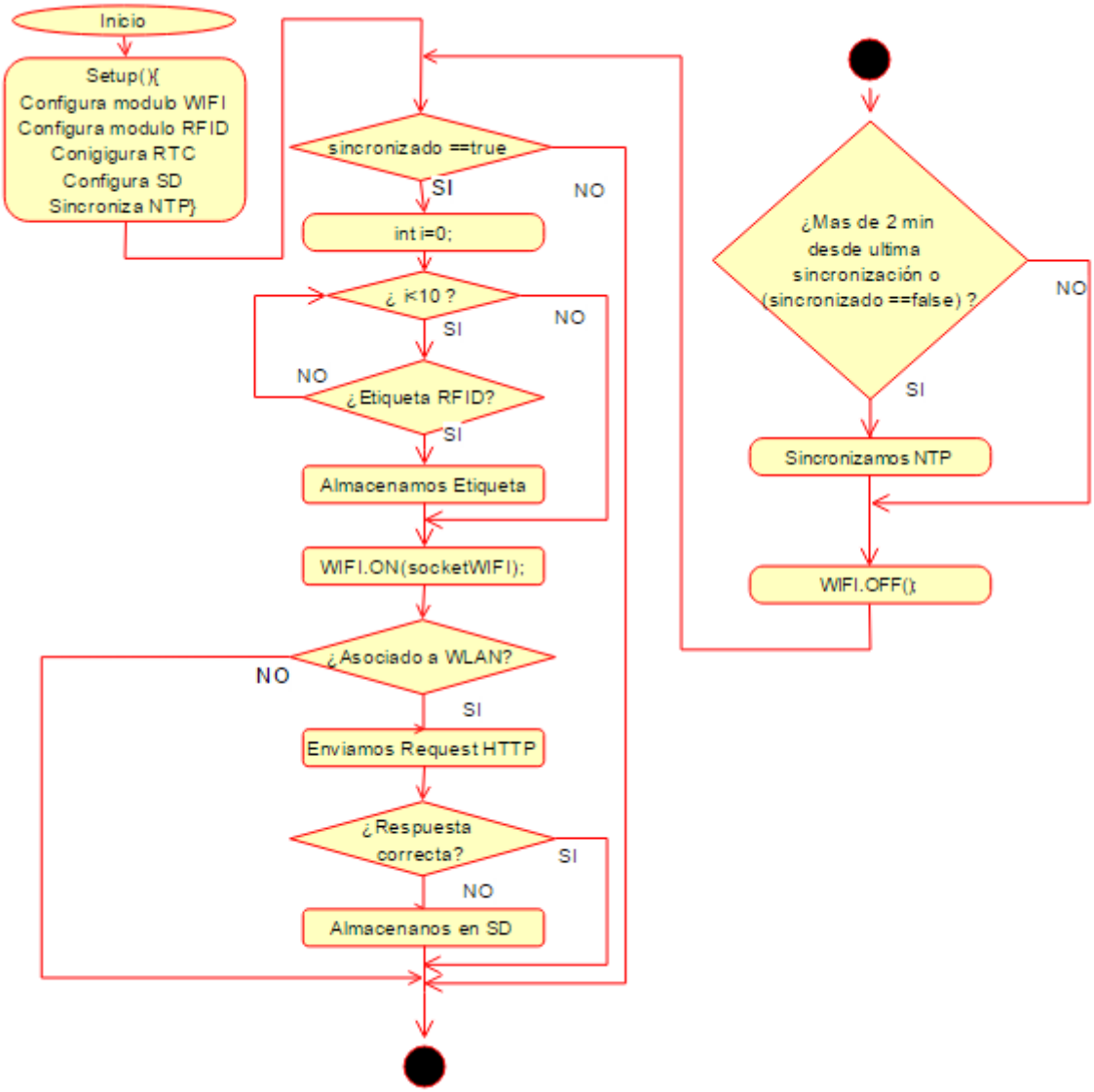

Figura 26. Diagrama de flujo del programa Waspmote

4.2.3 Diseño de la interfaz Web

En esta sección pretende recoge el diseño del que se ha partido para el desarrollo de la interfaz web.

Podemos dividir el diseño de la interfaz Web en:

- **Contenido:** Algunas de las nuevas etiquetas de HTML5 nos permiten diseñar un esqueleto mediante bloques sobre el que se distribuirá el contenido de la web. Aunque estas etiquetas no tienen ningún impacto en el estilo visual de la página nos serán de utilidad para diseñar un esquema. La Figura 27 representa el diseño utilizado para la página principal de este proyecto.

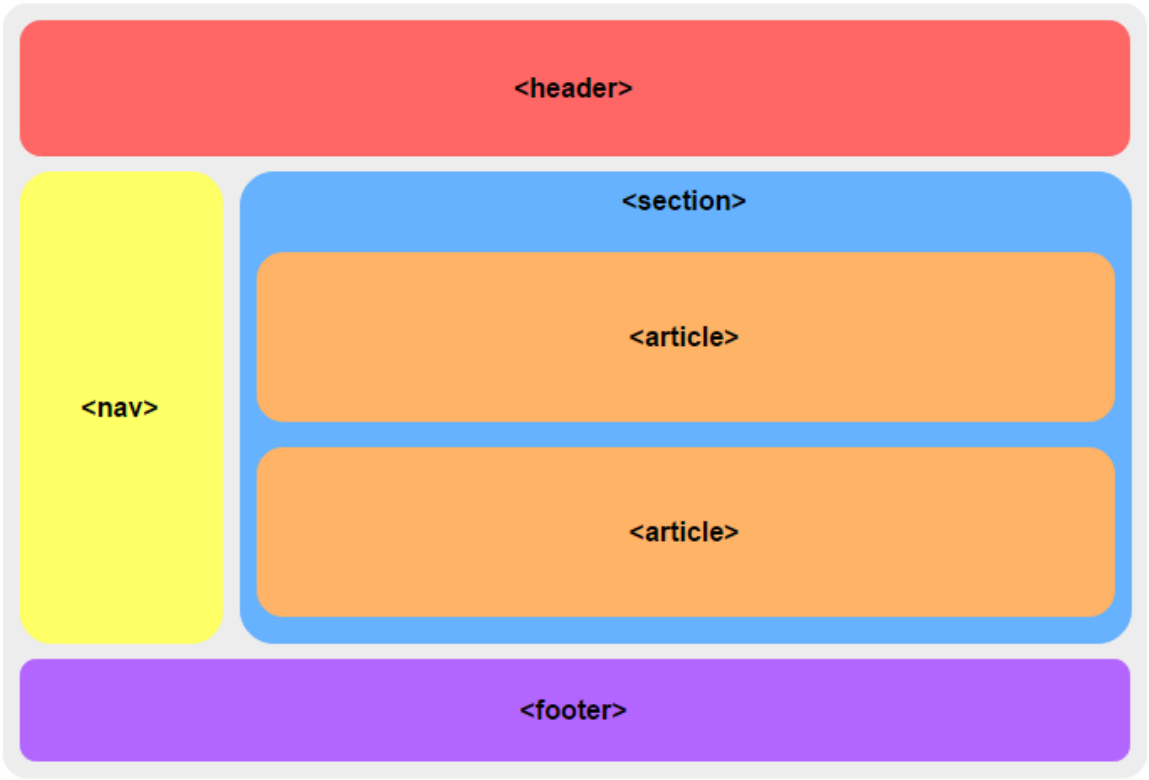

Figura 27. Diseño de la página principal de la interfaz web con HTML5

- **Estilo:** El diseño visual de la interfaz Web se recae en su totalidad sobre CSS3. La Figura 28 pretende reflejar el diseño objetivo de la interfaz de la pantalla principal del proyecto una vez se hay aplicado CSS3 a la interfaz web del sistema de control de inventario.

  La potencia de la tecnología de CSS3 ha permitido generar efectos dinámicos como menús desplegables y deslizantes que han permitido diseñar una interfaz simple e intuitiva.

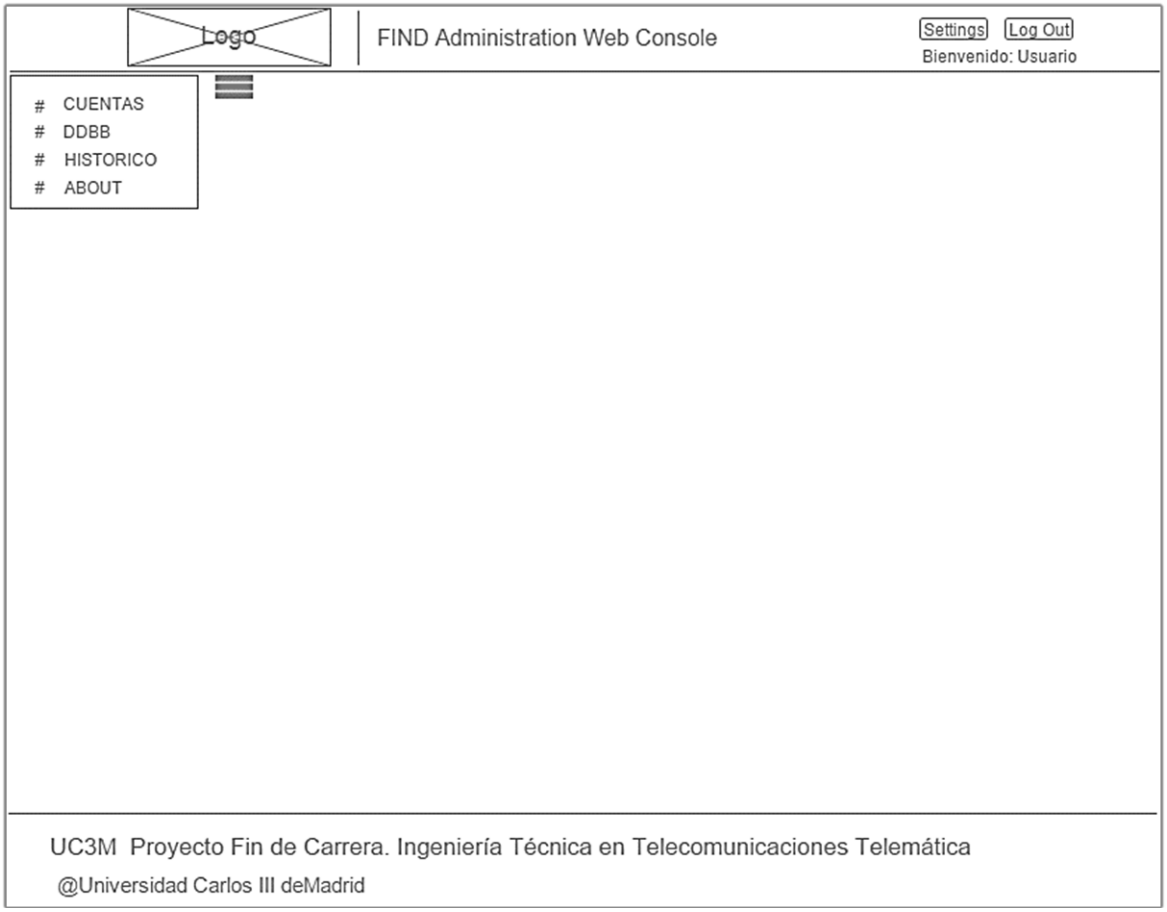

Figura 28. Diseño conceptual de la página principal de la WEB tras aplicar CSS.

4.2.4 Aplicaciones telemáticas

Algunos requisitos hacen referencia a la necesidad de la existencia de algunos servicios telemáticos como NTP o la capacidad del sistema de enviar correos. Por lo que, intentando dar una solución a estos requisitos en la fase de diseño del sistema se ha decido incluir en el servidor estos servicios. Los servicios incluidos son:

- **NTP:** El servidor del sistema tiene instalada la aplicación NTP daemon. Este servicio permitirá a los nodos Waspmote estar sincronizados.

- **SMTP:** Como se ha indicado, los requisitos solicitan que el sistema tenga la capacidad de enviar correos electrónicos automáticos. Aunque estos correos podrían enviarse de forma anónima, este tipo de soluciones siempre causan problemas al no contar con un servicio DNS con un dominio público. Por lo que se ha decido instalar un servidor de correo en el servidor del propio sistema y crear a los usuarios buzones locales al servidor. El SMTP instalado en el servidor es Postfix.

- **DNS:** Puesto que la aplicación de envío de correos automáticos consulta los registros MX del dominio de destino se hace necesario contar con un servidor DNS en el entorno al haber decidido utilizar un servidor SMTP en el sistema.

4.2.5 Diagrama del diseño del sistema

En el diseño final de todo el sistema nos encontramos con los siguientes elementos:

- **El sistema de control de inventario:** en el entorno servidor en el que se distinguen:
    - **Aplicación web:** El servicio web es utilizado para 2 fines:
        - Por los administradores para realizar operaciones sobre el sistema.
        - Para recibir las lecturas de los dispositivos Waspmote.
    - **Servicio NTP:** Utilizado por los nodos Waspmote para sincronizar su reloj con el servidor del entorno.
    - **Servicio DNS:** Dispone de los registros MX.
    - **Servicio SMTP:** Necesario para almacenar los buzones de los responsables y los administradores del entorno.
    - **Las bases de datos:** El entorno cuenta con la base de datos de administradores y la base de datos de negocio.
- **Waspmote:** Los nodos que realizan las lecturas para la realización del inventario.

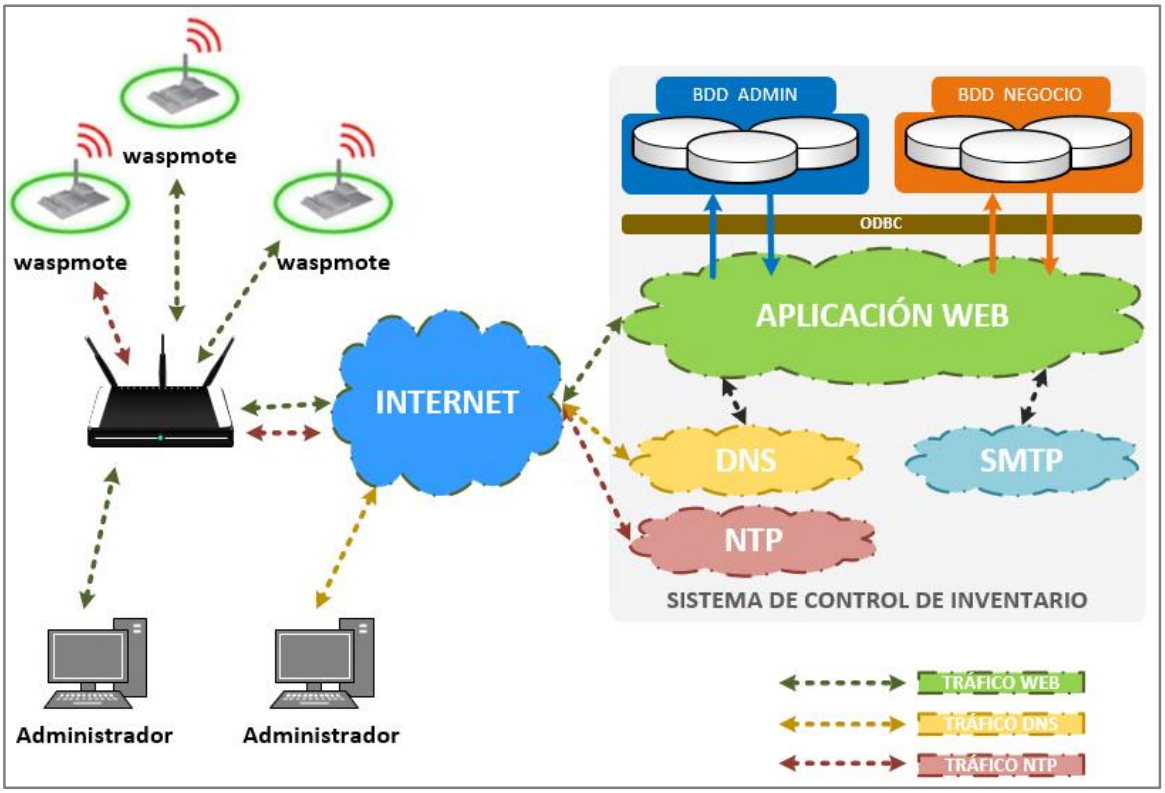

Figura 29. Diseño general del sistema de control de inventario

Capítulo 5

Desarrollo del Sistema de Control de Inventario

En este capítulo se describen los aspectos más relevantes de la fase de desarrollo. La implementación del código, el estilo de la interfaz web y la configuración de algunos elementos del entorno del sistema.

5.1 Servicios telemáticos

El funcionamiento del sistema de control de inventario requiere la existencia en el entorno de un conjunto de servicios. Para el proyecto estos servicios se han instalado en el mismo servidor de la aplicación web, por lo que se han instalado sobre el sistema operativo Ubuntu 12.04 LTS. Los servicios son:

- **NTP:** La instalación del demonio ntpd en el sistema operativo es muy sencilla, solo requiere ejecutar el comando.

```
sudo apt-get install ntp
```

El demonio no ha requerido ningún tipo de información adicional.

- **DNS:** Para ofrecer este servicio al entorno se ha instalado bind9. De nuevo la instalación de este servicio es muy sencilla, solo es necesario ejecutar el siguiente comando:

```
sudo apt-get install bind9
```

La instalación de este servicio ha requerido la creación de unos registros DNS para el entorno. Se ha utilizado el dominio **findbackend.local**, los registros DNS se encuentra en el Anexo II. Registros DNS.

- **SMTP:** Para ofrecer este servicio se ha instalado Postfix. La instalación de Postfix se ha realizado ejecutando el siguiente comando en una consola:

```
sudo apt-get install postfix
```

Para la configuración del servidor SMTP es necesario cambiar los parámetros mydestination y mynetworks en el fichero main.cf como se indica en la Figura 30.

```
mydestination = findbackend.local, localhost.localdomain, localhost
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 192.168.1.0/24
```

Figura 30. Configuración del servidor SMTP

5.2 Base de datos

El detalle de las tablas resultantes de la implementación de las bases de datos que se han modelado en los diagramas de la Figura 24 y la Figura 25 se puede encontrar en el Anexo III. Scripts de la Base de datos.

5.3 Waspmote

En esta sección se hablara de los detalles relativos a la implementación del código de la placa Waspmote.

5.3.1 Estructura del código

La implementación del código sigue, dentro de lo posible, la estructura recomendada por el fabricante y detallada en el punto 3.1.2.1.

Las partes del código se dividen en:

- En la primera parte se incluyen las librerías "WaspWIFI.h" y "WaspRFID13.h" y se definen las constantes y variables globales que se van a utilizar en el código (ver Figura 31).

```
////////////////////////////////////////////////////
// Inclusion de las librerias que se utilizan
#include <WaspWIFI.h>
#include <WaspRFID13.h>

//////////////////////CONSTANTS//////////////////////////
////////////////Datos de la WLAN/////////////////////
#define ESSID "WLAN_1"
#define AUTHKEY "WLAN_pass"
////////////////////////////////////////////////////
////////////////WEB server ip address/////////////
#define WEB_SERVER "192.168.1.34"
////////////////////////////////////////////////////
////////////////NTP server ip address/////////////
#define NTP_SERVER "192.168.1.34"
////////////////////////////////////////////////////
////////////////NTP server ip address/////////////
#define REMOTE_NTP_PORT 123
```

Figura 31. Librerías y constantes utilizadas en el código

- En la segunda parte, en la función setup(), se realiza la configuración inicial todos los módulos que van a ser utilizados en el programa. Para ello, se realizan llamadas a las funciones descritas en el punto 5.3.2 que se encargan de realizar dicha configuración.

- En la tercera parte, en la función loop(), se encuentra la implementación principal del código. La implementación de esta parte sigue el diagrama de flujo diseñado en el punto 4.2.2 (ver **¡Error! No se encuentra el origen de la referencia.**).

Cabe resaltar que, según las recomendaciones de Libelium, al finalizar el bucle y terminar la ejecución del programa se debe poner la placa en uno de los modos de bajo consumo. Dada la naturaleza del proyecto en la cual es necesario que el dispositivo esté preparado para hacer lecturas RFID en cualquier momento, y que no es posible que el módulo RFID despierte a la placa no hemos implementado ninguno de los modos de bajo consumo de la placa. Aun así siendo conscientes de que la idea a la hora de trabajar con nodos sensores es tener muy presente la autonomía del dispositivo para potenciar su eficiencia energética se ha optado por apagar el módulo WiFi al final de cada bucle y encenderlo solamente cuando se hayan producido lecturas en el módulo RFID y sea necesario enviar datos al servidor.

5.3.2 Funciones implementadas

Para poder completar la funcionalidad del programa desarrollado se ha hecho necesario implementar una serie de funciones. Las funciones implementadas son las siguientes.

- **wifiInitialConf():** Esta función realiza la siguiente configuración sobre el módulo wifi:
    - Enciende el módulo:
    - Activa el protocolo HTTP y UDP en el módulo.
    - Pone el modo de asociación a WLAN manual.
    - Añade la contraseña a la WLAN.
- **rfidIntialConf():** Enciende el módulo RFID.
- **rtcInitialConf():** Enciende el dispositivo RTC y añade el GMT correspondiente a la fecha actual.
- **setSNTP_RTC():** Sincroniza la hora con el servidor NTP y actualiza el RTC.
- **appendOffline():** Almacena en un fichero de la SD las request que han tenido algún error en su envío.
- **getStats():** Muestra las estadísticas de la placa desde que se ha encendido.

5.3.3 HTTP request

En la URL enviada al servidor es necesario incluir los siguientes datos:

- **Marca de tiempo**: Es necesario que los envíos al servidor tengan una fecha para que luego puedan ser organizados apropiadamente por el sistema de control de inventario.

- **UID**: Es el UID que identifica a la etiqueta leída.

- **Número de serie**: Para identificar el dispositivo remitente se envía el número de serie del dispositivo como en el mensaje HTTP.

- **MAC:** De nuevo, este dato tiene como objetivo identificar el dispositivo remitente del envío del mensaje.

  Para obtener la información de la MAC se ha implementado una función nueva en la librería WiFi de Waspmote a la que hemos llamado getMAC_custom(). Aunque la librería WiFi ya disponía de una función para obtener la MAC, esta función solo escribía el dato en la UART para poder ser observada en el "Serial Monitor" y en nuestro código necesitábamos el dato para operar con él.

Un ejemplo con el formato final de la URL utilizada es:

GET$/W?t=27_09_15_23_14_52&u=9E9CECEB&i=356904994&m=00066680f738

Donde el parámetro **t** representa la marca de tiempo en formato dd_MM_YY_HH_mm_ss, el parámetro **u** es el UID en formato hexadecimal, el parámetro **i** representa el número de serie de la placa y el parámetro **m** contiene la MAC de la interfaz en un formato sin puntos.

5.3.4 LED configurables

Dado que el RFID es un elemento que interactúa con el usuario hemos necesitado un método que le haga saber al usuario de la placa el momento en que la placa se encuentra disponible para realizar lecturas y cuando se encuentra ocupada en otras tareas. Siguiendo la recomendación de Libelium se ha hecho uso de los LEDs configurables para este fin.

Cuando el LED configurable verde se encuentra encendido le indica al usuario que el módulo RFID está disponible para realizar lecturas (ver Figura 32).

```
// LEDs indica el STATUS de RFID
// Green LED ON indica que RFID esta preparado para leer
// Red LED (LED0)  ------>  OFF
// Green LED (LED1)------>  ON
USB.println(F("----> Encendido LED Verde [Equipo disponible]"));
Utils.setLED(LED0, LED_OFF);
Utils.setLED(LED1, LED_ON);
```

Figura 32. Encendido del LED verde

Cuando el LED configurable rojo se encuentra encendido le indica al usuario que el módulo RFID se encuentra ocupado en otras tareas y no está disponible para realizar lecturas (ver Figura 33)

```
// LEDs indica el STATUS de RFID
// Red LED ON indica que RFID esta NO puede realizar lecturas
// Red LED (LED0)  ------>  OFF
// Green LED (LED1)------>  ON
USB.println(F("----> Encendido LED Rojo [Equipo ocupado]"));
Utils.setLED(LED0, LED_ON);
Utils.setLED(LED1, LED_OFF);
```

Figura 33. Encendido del LED rojo

5.4 Aplicación web

En el alcance de esta sección se encuentra todo lo relativo a la fase de desarrollo relacionado con la implementación del código de la interfaz web y las distintas librerías que utiliza el sistema de control de inventario. Así como, una breve reseña de la configuración del contenedor de Servlets Apache Tomcat.

5.4.1 Estructura de directorios

La aplicación web está organizada en la estructura de directorios descrita en la Figura 34.

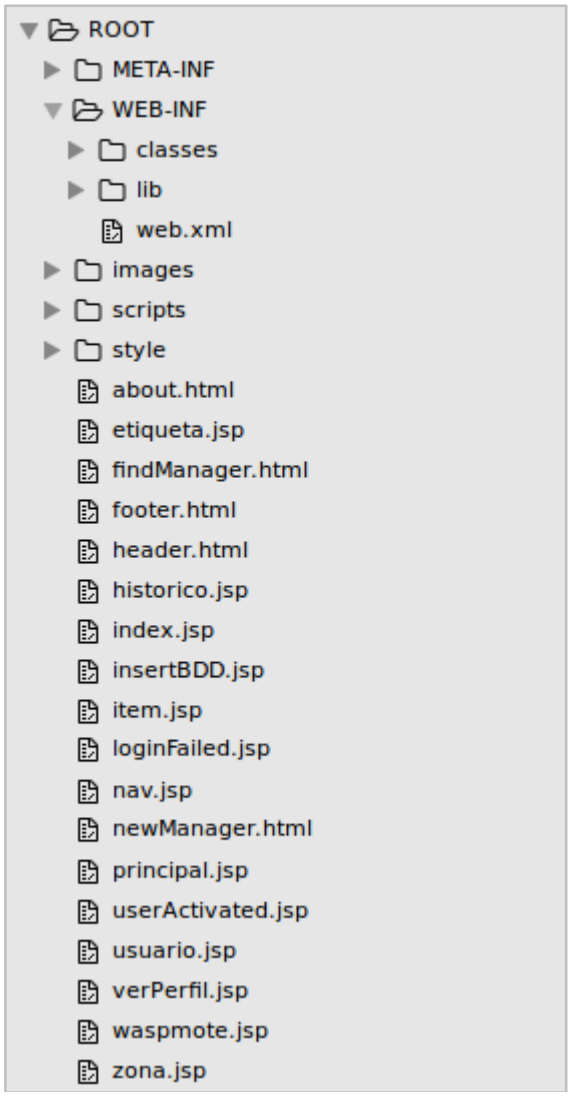

Figura 34. Estructura de directorios de la aplicación Web.

En la carpeta **ROOT** se encuentran:

- **JSPs y HTML:** Estos ficheros terminan definiendo el contenido de los ficheros HTML que recibe el cliente.

- La carpeta **WEB-INF** contiene:

    - Classes: En esta carpeta se encuentran los ficheros .class que ejecutaran la aplicación. En esta carpeta se encuentran los Servlets, los Beans y las librerías desarrolladas en el proyecto. En la Figura 35 se describe con más detalle el contenido de esta carpeta.

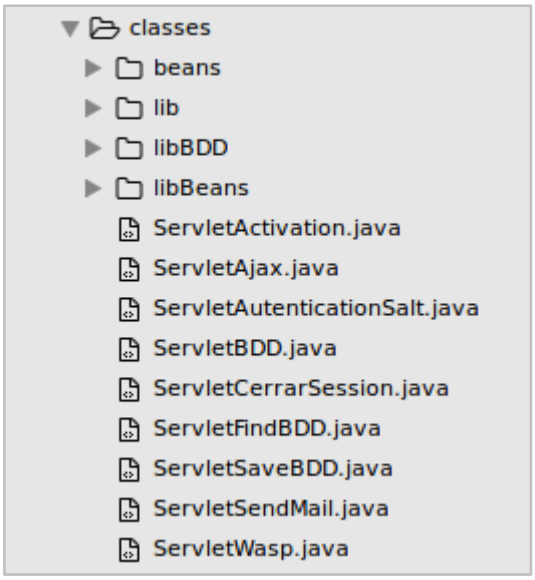

Figura 35. Estructura de directorios de la carpeta clases.

Los contenidos de la carpeta beans, lib, libBDD, libBeans serán descritos a lo largo del punto 5.4.3.

- Lib: Esta carpeta contiene los driver y librerías descritas en el punto 5.4.2.
- Web.xml: Este fichero contiene la configuración de la página y la definición de los servlets que se ejecutan en el entorno.

- La carpeta **images** contiene las imágenes que se utilizan en la página web.
- La carpeta **scripts** contiene los ficheros JavaScript que utiliza la página.
- La carpeta **style** contiene los ficheros css que definen el estilo final que tendrá la página.

5.4.2 Drivers y APIs

Para la implementación de la aplicación web se ha utilizado una serie de librerías java para poder llevar a cabo ciertas funcionalidades:

- **MySQL connector:** Es un driver que permite la conectividad para aplicaciones de java con una base de datos SQL. Este conector es una implementación del estándar para JDBC API.
- **Java Mail API:** Esta API se ha utilizado para implementar la librería que realiza el envío de correos electrónicos.

5.4.3 Implementación de la aplicación

En este punto se expone todo lo relativo a la implementación del código de la aplicación.

5.4.3.1 Modelo-Vista-Controlador del proyecto

Como se ha comentado la implementación de la aplicación web utiliza el patrón MVC:

- **Controlador:** El controlador está formado por los Servlets de la aplicación. A continuación se describen los Servlets que se han implementado en la aplicación web:

    - ServletActivation: Este Servlet recibe la petición de activación desde una URL que previamente ha sido enviada a una dirección de correo por la aplicación web.

      Cuando el Servlet recibe la petición HTTP comprueba que han pasado menos de 48 horas desde el envío del mail accediendo al registro donde se guarda la información en dicha URL en la tabla ManagerStatusURL. De ser así, activa el usuario administrador y elimina el registro de la tabla ManagerStatusURL Si han pasado más de 48 horas elimina simplemente la entrada de la tabla ManagerStatusURL.

    - ServletAjax: Este Servlet es el encargado de atender todas las consultas realizada mediante Ajax de la consola de administración web y enviar la respuesta.

    - ServletAutenticationSalt: Este Servlet comprueba los datos de login de la consola web. Para hacerlo utiliza la librería EncryptDecryptLib del paquete lib y descrita en el punto 5.4.3.2.

    - ServletBDD: Este Servlet recibe todos los datos introducidos por formularios y comprueba la integridad de los datos. A continuación comprueba si es una operación de inserción o de búsqueda. Si es una operación de inserción lo pasa al Servlet ServletSaveBDD y si es una operación envía la request al Servlet ServletFindBDD.

    - ServletCerrarSesion: Este Servlet invalida la sesión existente con el servidor.

    - ServletFindBDD: Este Servlet se encarga de realizar las búsquedas procedentes de los formularios de la aplicación web sobre ambas bases de datos. Para ello hace uso de las librerías del paquete libBDD. El paquete libBDD se describirá en el punto 5.4.3.2.

- ServletSaveBDD: Este Servlet ejecuta las operaciones de inserción y actualización procedentes de los distintos formularios de la aplicación web. Para realizar estas operaciones utiliza las librerías del paquete libBDD.

- ServletSendMail: Este Servlet recibe las request que implican el envío de e-mails. Para el envío de los correos electrónicos hace uso de la librería MailLib del paquete lib.

  El email enviado se envía en formato HTML al buzón del usuario (ver Figura 36).

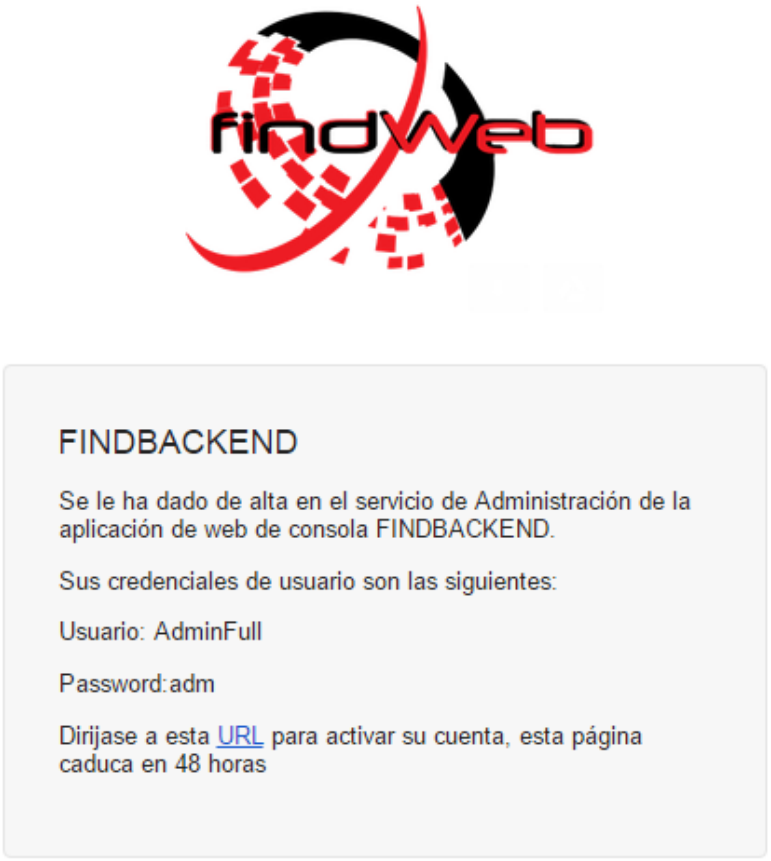

Figura 36. Formato del email de activación

- ServletWasp: Este Servlet es el encargado de gestionar las request HTTP procedentes de los nodos Waspmote.

- **Modelo**: El modelo hace referencia a los datos que maneja la aplicación. En este caso los beans representan las distintas tablas de la base de datos.

  Los distintos beans de este punto conforman el paquete **beans**. Este paquete se localiza en la estructura de directorios dentro de la carpeta "classes".

  Los beans implementados en este paquete en la solución final son los siguientes:

- Historic: Define el bean que representa a la tabla Historic de la base de datos Negocio. En la Figura 37 se muestran los atributos que se han implementado en el bean.

```
//beans que implementa la tabla de historicos
public class Historic implements java.io.Serializable{

    //id del registro
    private int idHistoric;
    //Fecha del registro
    private Date dateReg;
    //Identificador de tarjeta UID
    private Long uid;
    //MAC del dispositivo Waspmote
    private String waspmoteMAC;
    //Serial del dispositivo Waspmote
    private String waspmoteSerialID;
    //Id de la zona donde se supone se esta leyendo la RFID
    private int idZone;
```

Figura 37. Atributos del bean Historic

- Item: Define el bean correspondiente a la tabla Items de la base de datos Negocio. La Figura 38 contiene los distintos atributos que posee el bean.

```
//beans que implementa la tabla Items
public class Item implements java.io.Serializable{


    //identificador unico de producto
    private int IdItem;
    //identificador del responsable
    private int IdResponsable;
    //Numero de serie del producto
    private String NumSerie;
    //Fecha de alta
    private Date RegistrationDate=null;
    //Fecha de baja si la hay
    private Date TerminationDate=null;
    //Descripcion del producto
    private String Description;
```

Figura 38. Atributos del bean Item

- Responsable: Este bean implementa la tabla Responsable de base de datos de Negocio. En la Figura 39 se pueden observar los distintos atributos del bean.

```
//beans que implementa la tabla responsable
public class Responsable implements java.io.Serializable{


    //identificador unico del responsable
    private int IdResponsable;
    // nombre y apellidos del responsable
    private String Name;
    private String Surname;
    //Cargo que desempeña en el entorno
    private String Charge;
    //email del usuario
    private String Email;
```

Figura 39. Atributos del bean Responsable

- Tag: Este bean implementa la tabla Etiquetas de la base de datos Negocio en el entorno de la aplicación. La Figura 40 muestra los atributos del bean.

```
//beans que implementa la tabla etiquetas
public class Tag implements java.io.Serializable{


    //UID identificado unido de etiqueta
    private Long UID;
    //tipo de etiqueta "1" de item "0" de zona
    private String Type;
```

Figura 40. Atributos del bean Tag

- TagItem: Con este bean se modela la tabla EtiquetasItem de la base de datos Negocio. Los atributos del bean se observan, a continuación, en la Figura 41.

```
//beans que implementa la tabla EtiquetasItem
public class TagItem implements java.io.Serializable{

    //identificador unico de Tarjeta RFID
    private Long UID;
    //zona donde se encuentra la tarjeta.
    private int IdZone;
    //Id del producto
    private int IdItem;
```

Figura 41. Atributos del bean TagItem

- TagZone: La tabla EtiquetasZona es modelada en la aplicación mediante este bean. En la Figura 42 se muestran los atributos de la clase.

```
//beans que implementa la tabla EtiquetasZona
public class TagZone implements java.io.Serializable{

    //identificador unico de Tarjeta RFID
    private Long UID;
    //identificado de zona donde se encuentra la tarjeta.
    private int IdZone;
```

Figura 42. Atributos del bean TagZone

- Waspmote: Este bean modela la tabla Waspmote de la base de datos Negocio. La Figura 43, a continuación muestra los atributos del bean.

```
//beans que implementa la tabla Waspmote
public class Waspmote implements java.io.Serializable{



    private int idWapmote;
    //Mac del dispositivo
    private String mac;
    //Identificado de dispositivo
    private String deviceID;
    //Status del dispositivo
    private Boolean status;
```

Figura 43. Atributos del bean Waspmote

- Zone: Este bean implementa en la aplicación web la tabla Zonas de la base de datos Negocio. En la Figura 44 se observan los atributos que posee el bean.

```
//beans que implementa la tabla Zonas
public class Zone implements java.io.Serializable{

    //Id de la zona
    private int IdZone;
    //Id del Responsable
    private int IdResponsable;
    //Edificio
    private String Building;
    //Piso
    private String Floor;
    //letra
    private String Letter;
    //oficina
    private String Room;
    //Departamento
    private String Department;
```

Figura 44. Atributos del bean Zone

- Manager: Este bean implementa en la web la tabla Manager de la base de datos Admin. En la Figura 45 se pueden ver los atributos que posee la clase.

```java
//beans que implementa la tabla manager
public class Manager implements java.io.Serializable{


    //identificador unico del usuario administrador
    private int idManager;
    //alias de administrador en el sistema
    private String alias;
    //Salt y Hash de la password de usuario
    private String salt;
    private String hash;
    //nombre y apellidos del usuario
    private String name;
    private String surname;
    //Nivel de permiso en la aplicacion "FULL ADMIN", "BDD ADMIN","REPORT ADMIN"
    private String rol;
    //envio de correos
    private String email;
    //activo o inactivo
    private Boolean status;
```

Figura 45. Atributos del bean Manager

- ManagerStatusURL: Esta clase modela la tabla ManagerStatusURL de la base de datos Admin. En la Figura 46 se muestran sus atributos.

```java
//beans que implementa la URL de activación de usuario
public class ManagerStatusURL implements java.io.Serializable{


    //identificador unico de la URL
    private int idStatusURL;
    //URL
    private String url;
    //fecha
    private Date date;
    //identificador unico del administrador al que aplica la URL
    private int idManager;
```

Figura 46. Atributos del bean ManagerStatusURL

- **Vista:** En este caso, dentro del patrón MVC, la vista está representada por los distintos JSPs y HTML que se han implementado en la aplicación web.

  Aunque existen diversos documentos HTML y JSP todos, excepto el index.jsp, son incluidos en el documento principal.jsp. Este documento es el encargado de mostrar al usuario la interfaz web y de recoger todas las consultas y respuestas que la aplicación permite.

5.4.3.2 Librerías implementadas

Durante el desarrollo el proyecto se han implementado las siguientes librearías:

- Librerías de bases de datos: En el directorio “classes” se encuentra el paquete **libBDD**. Esta librería contiene los métodos para interactuar con las base de datos del sistema. Las clases que componen la librería son:

    - InterfaceBDD: Esta clase tiene los métodos que permiten realizar operaciones de inserción, búsqueda, borrado y actualización sobre las tablas de la base de datos Negocio.

    - InterfaceWebBDD: Esta clase tiene los métodos que permiten realizar operaciones de inserción, búsqueda, borrado y actualización sobre las tablas de la base de datos Admin.

- Librería de utilidades: De nuevo, en el directorio “classes”, se encuentra el paquete de utilidades llamado **lib**. Las clases que componen esta librería son:

    - EncryptDecryptLib: Esta clase contiene los métodos necesarios para almacenar las contraseñas encriptadas en la base de datos.

    - MailLib: Esta clase contiene los métodos para el envío de correos electrónicos.

5.4.3.3 Beans adicionales

Durante la implementación del código del sistema de control de inventario se han desarrollo algunos beans que no representan ninguna tabla de la base de datos. Estos bean se encuentra dentro del paquete **libBeans** son:

- SQLResponse: Este bean contiene el estatus de la respuesta de una operación realizada en una de las bases de datos. Además, si durante la operación con la base de datos se produce algún error los atributos de la clase también almacenan el código y el mensaje de la excepción producida. La Figura 47 muestra los atributos del bean.

```
//beans que implementa los errores que puede sufrir una insercción a la base de Datos
public class SQLResponse{

    //Booleano de respuesta
    private Boolean AnswerStatus;
    //mensaje de una excepcion sql
    private String SQLException;
    //mensaje de estado
    private String SQLState;
    //codigo de error
    private String ErrorCode;
```

Figura 47. Atributos del bean SQLResponse

- Mail: Los atributos de este bean modelan los parámetros de un correo electrónico. La Figura 48 muestra los atributos de la clase.

```
//beans que implementa un correo
public class Mail implements java.io.Serializable{

    //asunto del correo
    private String subject;
    //to del correo
    private String toEmail;
    //from del correo
    private String fromEmail;
    //from visible por el remitente
    private String fromName;
    //protocolo del correo
    private String protocol;
    //host de origen
    private String hostFrom;
    //dominio mx de los correos
    private String mxDomain;
    //puerto de destino
    private String port;
    //caracteristicas el correo
    private Properties properties;
```

Figura 48. Atributos del bean Mail

5.4.3.4 Inclusión de CSS3

En la última parte del desarrollo de la web se ha puesto esfuerzo en alcanzar un diseño agradable y sencillo para la interfaz.

El la Figura 49 se muestra el resultado en el diseño de la interfaz.

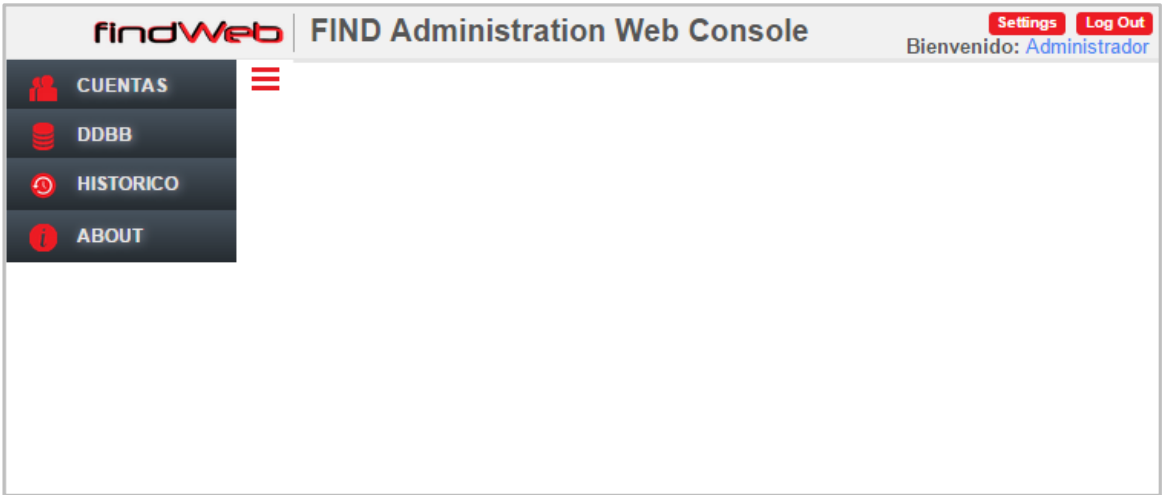

Figura 49. Apariencia de la ventana principal de la interfaz web.

5.4.3.5 Diagrama de clases de la implementación

En este punto se muestran los diferentes diagramas de clases utilizados en la implementación de los Servlets del sistema. La creación de estos diagramas no sigue una definición formal y la inclusión en el documento tiene una finalidad puramente descriptiva de la organización de los distintos Servlets.

Como se ha comentado el ServletBDD recibe la request procedente de todos los formularios de usuario, en la Figura 50 se define el diagrama de este caso. En la figura las flechas discontinuas de la imagen indican el uso de los distintos paquetes por los Servlets. Y las flechas continuas indican los distintos forward de la request que realizan los Servlets.

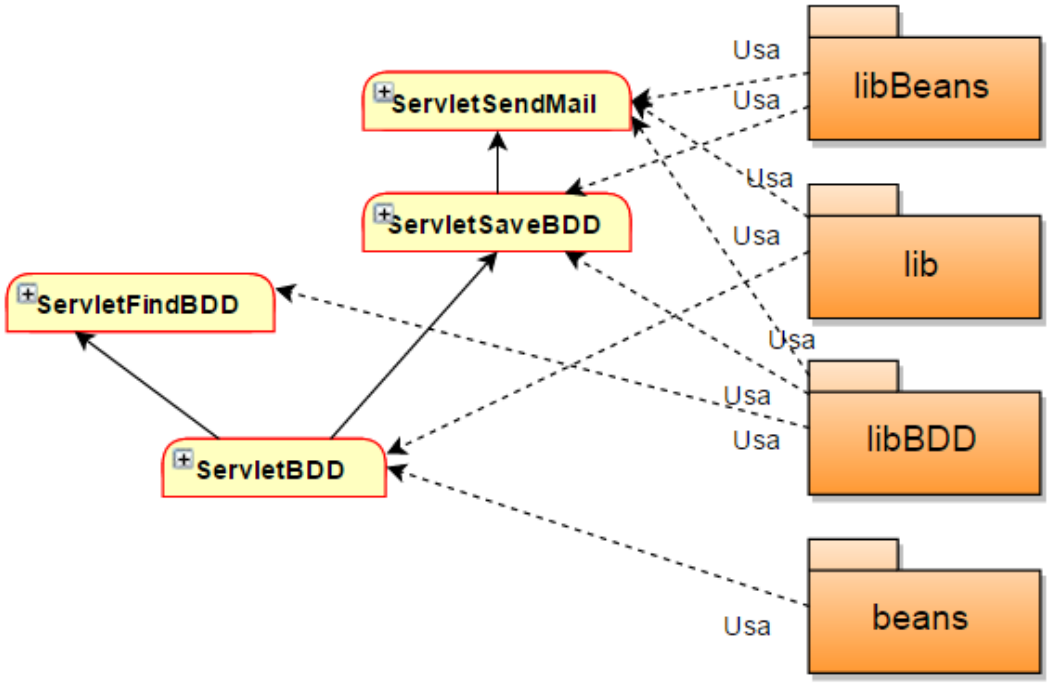

Figura 50. Diagrama de clases relacionadas con la interacción con las bases de datos

El ServletWasp recibe la request procedente de todos los dispositivos Waspmote la Figura 50 define el diagrama de este caso.

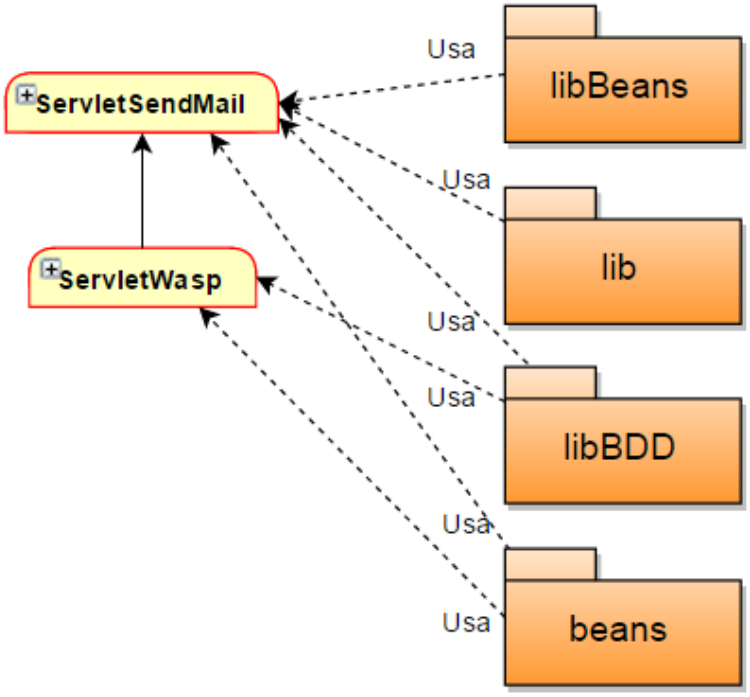

Figura 51. Diagrama de clases relacionadas con los datos recibidos de los nodos

El ServletAjax recibe todas las request que se realizan utilizando la tecnología AJAX. Este Servlet utiliza todos los paquetes que se han implementado en la solución final (ver Figura 52).

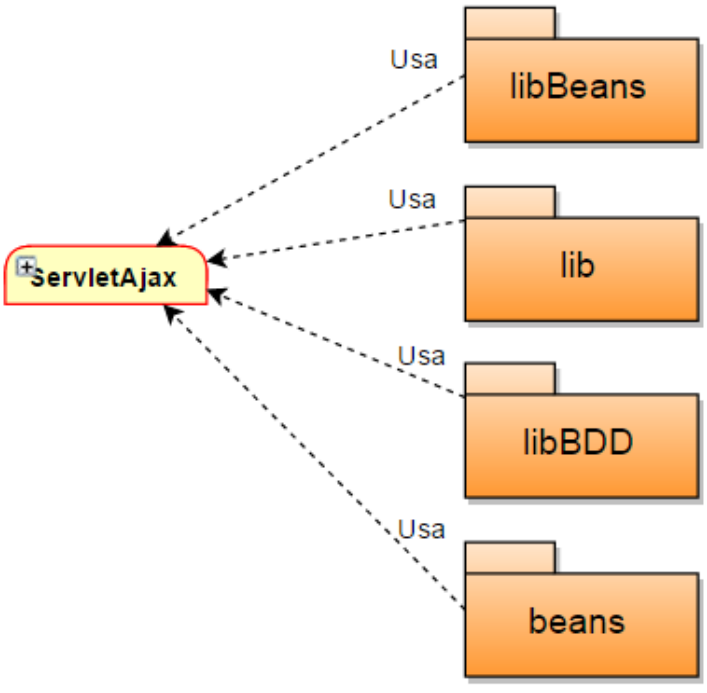

Figura 52. Diagrama de clases relacionadas con los datos enviados usando AJAX

El ServletAutenticationSalt recibe las request de la autenticación de usuarios.

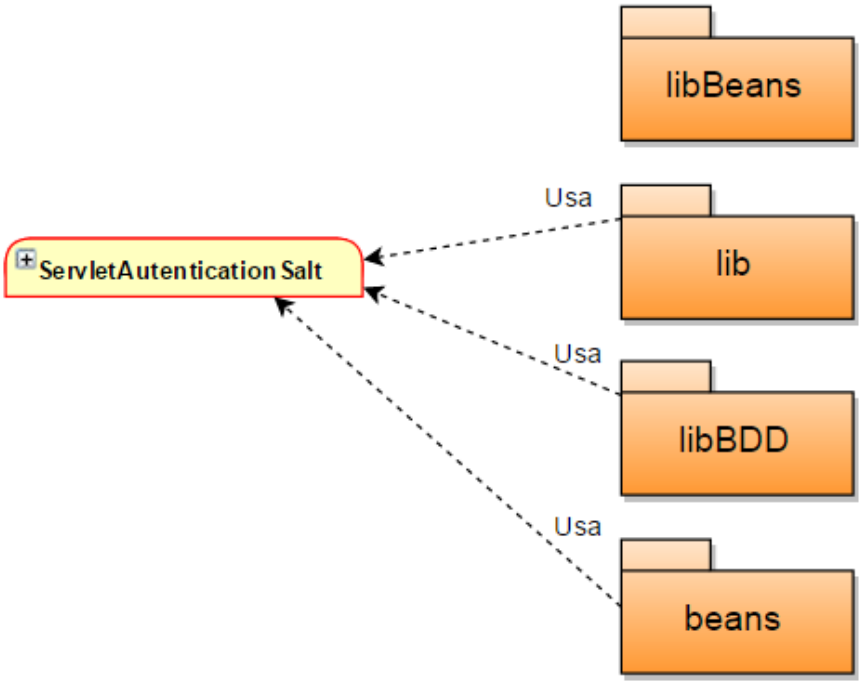

Figura 53. Diagrama de clases relacionadas con la autenticación de usuarios

El ServletActivation es el encargado de recibir la request de activación de los usuarios desde el mail de activación enviado al buzón del administrador.

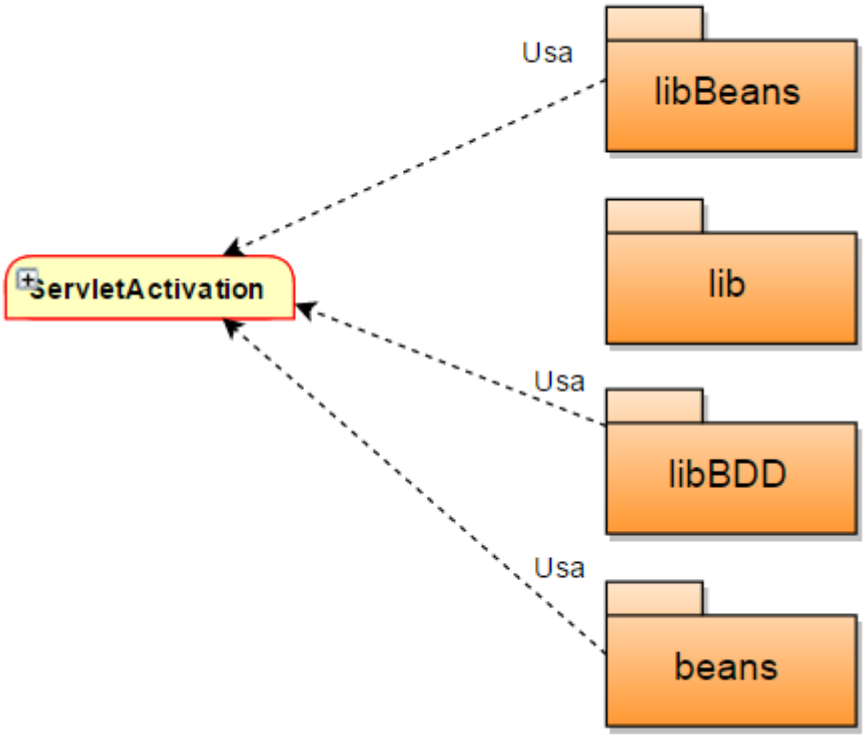

Figura 54. Diagrama de clases relacionadas con la activación de usuarios administradores

5.5 Conclusiones del capítulo

Este capítulo nos ha permitido exponer la implementación del sistema de control de inventario llevada a cabo en el proyecto. Como se indica en el presente capítulo la implementación del proyecto se ha realizado siguiendo los distintos puntos del diseño descritos en el capítulo "Análisis y Diseño".

Con la intención de no alargar este capítulo de manera innecesaria. Los detalles de la funcionalidad de la interfaz web se encuentran en el Anexo IV. Manual de usuario.

A continuación en el capítulo siguiente, "Pruebas", se valida la funcionalidad del sistema implementado.

Capítulo 6

Pruebas

Este capítulo recoge el conjunto de pruebas de aceptación para verificar el funcionamiento del sistema desarrollado. Para ello en algunos casos se recogen evidencias de la funcionalidad para demostrar que la solución implementada resuelve la cuestión planteada a lo largo del documento.

Se ha diseñado una batería de pruebas para probar la funcionalidad de los nodos Waspmote del entorno y otra para probar la funcionalidad del sistema de control de inventario completo.

6.1 Pruebas Waspmote

Este primer punto en el apartado de pruebas pretende mostrar la funcionalidad descrita del nodo Waspmote a lo largo de esta documentación. Para ello se ha probado la funcionalidad de:

- La sincronización NTP.
- La realización de lecturas RFID.
- El envío de datos mediante HTTP en distintas situaciones.

6.1.1 Sincronización NTP

El objetivo de esta prueba es comprobar la funcionalidad correcta de la sincronización de los nodos con el servidor.

Como se ha comentado con anterioridad el servidor del sistema tiene instalado el servicio NTP. La Figura 55 muestra la utilidad de NTPDaemon corriendo en el equipo servidor del sistema de control de inventario.

```
pablo@pablo-virtual-machine:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether 00:0c:29:c6:d2:ba brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.34/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec6:d2ba/64 scope link
       valid_lft forever preferred_lft forever
pablo@pablo-virtual-machine:~$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+static-21.herco 14.234.237.209   2 u  795  512  376   32.049    0.246   2.331
+dns01.masbytes. 158.227.98.15    2 u  123  512  377   37.510    0.902   3.181
-dnscache-madrid 140.203.204.77   2 u  441  512  377    4.236    5.237   3.551
*juniperberry.ca 193.79.237.14    2 u  384  512  377   37.727    0.611   3.201
```

Figura 55. Utilidad ntp corriendo en el servidor del sistema.

La placa Waspmote tras cargar los primeros módulos realiza una consulta NTP al servidor del entorno del sistema. La Figura 56 muestra los comandos que la librería WiFi ejecuta sobre el chip Wifly para realizar una consulta NTP contra el servidor del entorno.

```
set t a 192.168.1.34
set t p 123
set t e 1
time
----> Hora configurada desde el servidor NTP 192.168.1.34 el formato de fecha es  [Dia de la Semana YY/MM/DD, hh:mm:ss]
--------> La hora es: Mon, 15/09/21, 21:41:07
```

Figura 56. Salida por la interfaz "Serial Monitor" del IDE de la sincronización NTP con el servidor del entorno.

La Figura 57 muestra la comunicación del protocolo NTP entre el nodo Waspmote y el servidor del servicio.

```
54 18.9038960 192.168.1.39    192.168.1.34    NTP     90 NTP Version 3, client
55 18.9047570 192.168.1.34    192.168.1.39    NTP     90 NTP Version 3, server
```

Figura 57. Captura de la consulta y respuesta NTP contra el servidor del sistema

6.1.2 Lectura RFID

El objetivo de esta prueba es comprobar que el módulo RFID realiza las lecturas de la manera descrita en la documentación del proyecto.

Tras el arranque y la sincronización de la tarjeta, el módulo espera la lectura de una tarjeta RFID. Para indicar este estado se enciende el LED verde configurable de la placa (ver Figura 58).

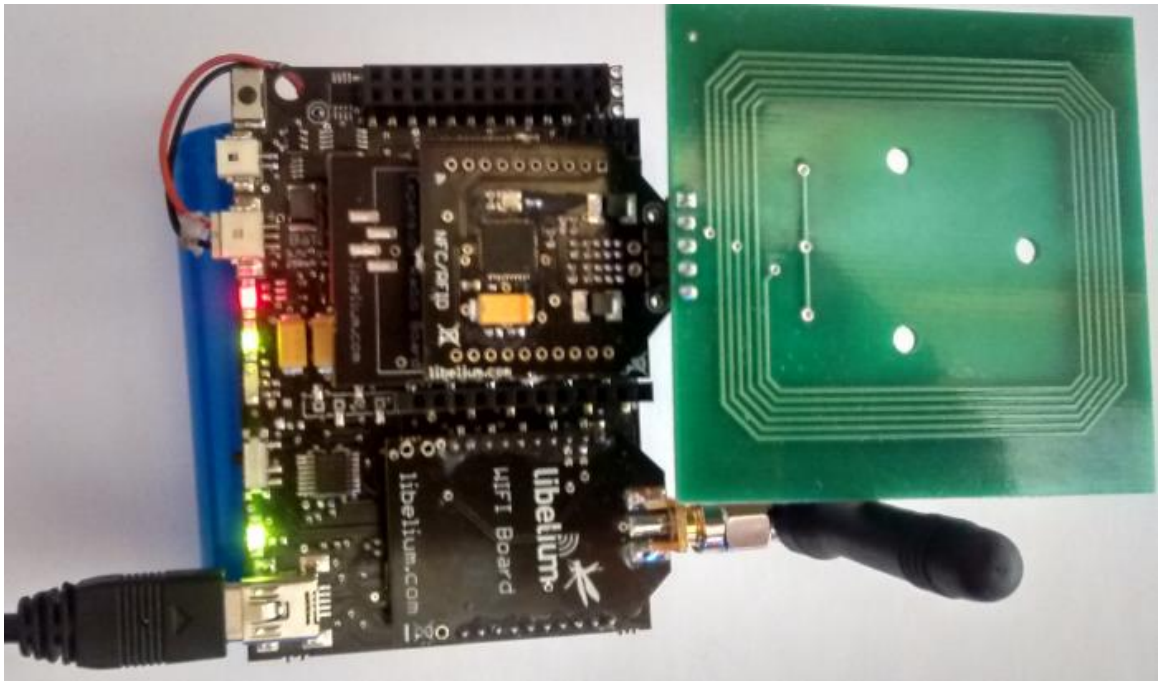

Figura 58. LED configurable verde encendido.

Cuando la tarjeta entra en el campo magnético el chip RFID obtiene el UID de la tarjeta (ver Figura 59)

```
----> RFID Ready to read
----> Encendido LED Verde [Equipo disponible]
--------> Tarjeta detectada: 0, Num. Bucle: 0

----> RFID Card Detected
--------> The UID: FE 05 EC EB

----> Parpadeo LED Verde
----> Encendido LED Rojo [Equipo ocupado]
```

Figura 59. Salida por la interfaz "Serial Monitor" del IDE tras la lectura de una tarjeta RFID.

Tras la lectura el dispositivo enciende el LED rojo configurable del dispositivo para indicar que la placa se encuentra ocupada y el módulo RFID no se encuentra esperando ninguna lectura (ver Figura 60).

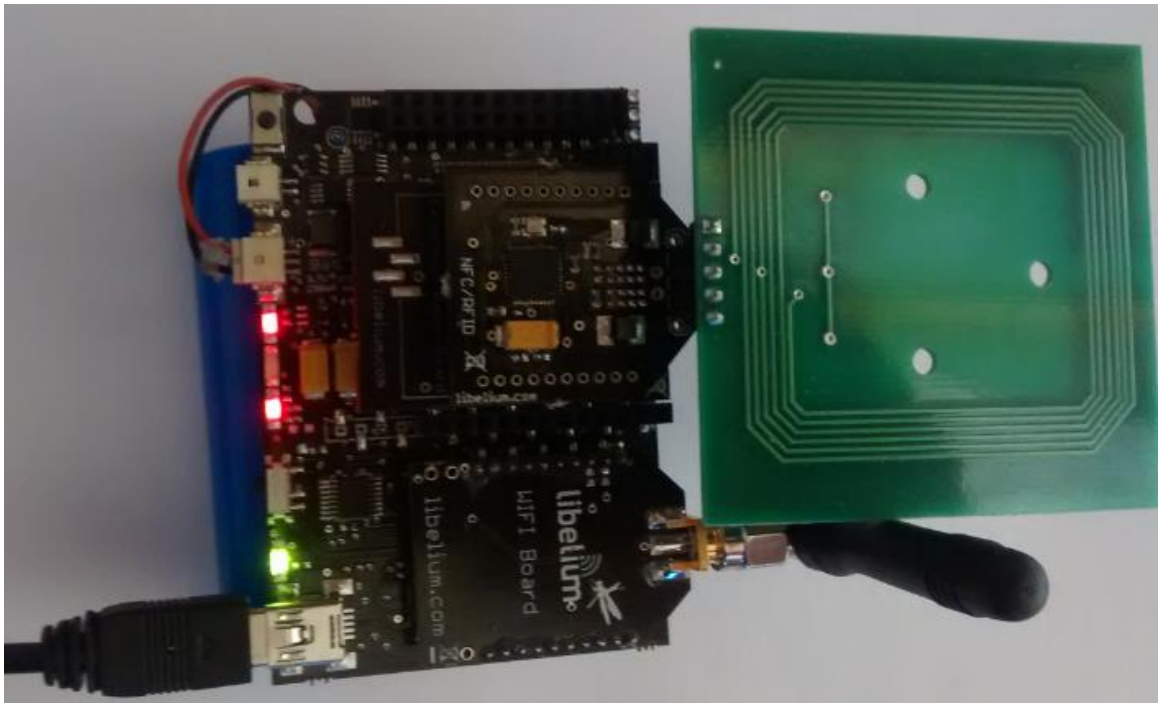

Figura 60. LED configurable rojo encendido.

6.1.3 Envío de datos mediante HTTP

El objetivo de las pruebas aquí descritas es comprobar el envío de datos en distintas situaciones.

6.1.3.1 Envío exitoso

Esta prueba parte de un estado en el que todas las partes del entorno están funcionando correctamente.

Como se ha comentado anteriormente los módulos Waspmote realizan el envío de los datos utilizando el protocolo HTTP. La Figura 61 y la Figura 62 muestran la funcionalidad del envío de datos utilizando HTTP contra el servidor web.

La Figura 61 muestra la salida por la interfaz “Serial Monitor del IDE”. En la figura podemos observar:

- La formación de la cadena de envío utilizando:

- La MAC y el número de serie del equipo para identificar del dispositivo.
- El UID que es la información leída en el dispositivo RFID y que identifica la tarjeta.
- La fecha de la lectura en forma de marca de tiempo.

- Los comandos que se ejecutan sobre el chip WiFi para establecer la sesión HTTP con el servidor.

- La respuesta del servidor con el código 200 OK. Código estándar para respuestas positivas.

```
----> Preparamos variables para la Request HTTP
--------> Variable 1: UID de tarjeta detectadaFE05ECEB
--------> Variable 2: Serial del dispositivo: 356904994
--------> Variable 3: MAC del dispositivo [formato sin puntos]00:06:66:80:f7:38
00066680f738

----> Encendido LED Rojo [Equipo ocupado]

----> El formato de la cadena Request con marca de tiempo es:GET$/W?t=22_09_15_18_31_41&u=FE05ECEB&i=356904994&m=00066680f738
set i h 192.168.1.34
set i r 80
set c r GET$/W?t=22_09_15_18_31_41&u=FE05ECEB&i=356904994&m=00066680f738
set o f 1
set u m 0
open
open
<2.32> *OPEN*
R: HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 7
Date: Tue, 22 Sep 2015 16:31:49 GMT
Connection: close

OK|true*CLOS*
enter CMD at 115200

----> HTTP query OK.
WIFI.answer:
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 7
Date: Tue, 22 Sep 2015 16:31:49 GMT
Connection: close

OK|true*CLOS*
----> Encontramos la marca de respuesta positiva en WIFI.answer
*OFF
**************************
```

Figura 61. Salida por la interfaz "Serial Monitor" del IDE de la sesión TCP.

La Figura 62 muestra el detalle de toda la sesión TCP establecida. En la imagen podemos distinguir:

- El "three-way handshake" para establecer la sesión TCP.
- El envío del comando GET para el envío de los datos al servidor.
- La respuesta 200 OK del servidor.
- El fin de la sesión TCP con el "four-way handshake".

49	22.280742	192.168.1.39	192.168.1.34	TCP	60 26587→80 [SYN] Seq=0 Win=3072 Len=0 MSS=1500
50	22.281281	192.168.1.34	192.168.1.39	TCP	58 80→26587 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
51	22.285968	192.168.1.39	192.168.1.34	TCP	60 26587→80 [ACK] Seq=1 Ack=1 Win=3072 Len=0
61	23.582720	192.168.1.39	192.168.1.34	HTTP	143 GET /W?t=22_09_15_18_31_41&u=FE05ECEB&i=356904994&m=00066680f738 HTTP/1.0
62	23.583277	192.168.1.34	192.168.1.39	TCP	54 80→26587 [ACK] Seq=1 Ack=90 Win=29200 Len=0
63	23.799425	192.168.1.34	192.168.1.39	HTTP	226 HTTP/1.1 200 OK (text/html)
64	23.800452	192.168.1.34	192.168.1.39	TCP	54 80→26587 [FIN, ACK] Seq=173 Ack=90 Win=29200 Len=0
65	23.802510	192.168.1.39	192.168.1.34	TCP	60 26587→80 [ACK] Seq=90 Ack=173 Win=3072 Len=0
66	23.804166	192.168.1.39	192.168.1.34	TCP	60 26587→80 [FIN, ACK] Seq=90 Ack=174 Win=3071 Len=0
67	23.817256	192.168.1.34	192.168.1.39	TCP	54 80→26587 [ACK] Seq=174 Ack=91 Win=29200 Len=0

Figura 62. Captura de la comunicación de la sesión TCP entre la placa y el servidor.

6.1.3.2 Envío erróneo

Esta prueba parte de un estado en el que el servido HTTP del servidor se encuentra parado. Figura 63

```
pablo@pablo-virtual-machine:~$ sudo service tomcat7 stop
 * Stopping Tomcat servlet engine tomcat7                    [ OK ]
```

Figura 63. Muestra de la parada del servicio

Las Figura 64 y Figura 65 muestran el error obtenido ante la imposibilidad de establecer la sesión HTTP.

```
set i h 192.168.1.34
set i r 80
set c r GET$/W?t=27_09_15_23_14_52&u=9E9CECEB&i=356904994&m=00066680f738
set o f 1
set u m 0
open
openHTTP failed.ERR entering CMD at 115200
*OFF
ERR: $$$
CMD not entered

----> HTTP ERROR

----> Ha existido algun problema en la comunicacion
--------> El fichero ya existe
--------> Almacenamos la request en la SD
*OFF
```

Figura 64. Salida por la interfaz “Serial Monitor” del IDE del error HTTP.

192.168.1.39	192.168.1.34	TCP	60 51189→80 [SYN] Seq=0 Win=3072 Len=0 MSS=1500
192.168.1.34	192.168.1.39	TCP	54 80→51189 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Figura 65. Captura de la sesión TCP fallida

En esto casos el programa desarrollado para la placa debe de salvar en un fichero la request HTTP realizada en la tarjeta SD para que los datos no se pierdan tal como se muestra en la Figura 66.

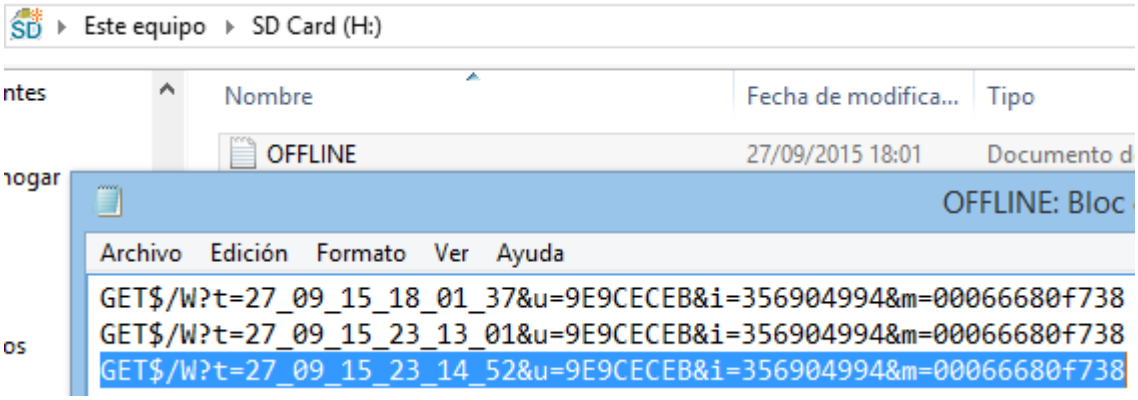

Figura 66. Request HTTP almacenada en tarjetas SD

6.1.3.3 Envío con el status deshabilitado

Esta prueba es una particularidad de la prueba anterior en la que la placa no se encuentra habilitada en el entorno para el envío de datos.

Si en la tabla Waspmote de la base de datos el registro correspondiente a una tabla tiene el campo STATUS deshabilitado los datos reportados por esa placa no serán aceptados por el sistema.

La Figura 67 muestra la lista obtenida por medio de la interfaz Web de dispositivos Waspmote deshabilitados en el sistema.

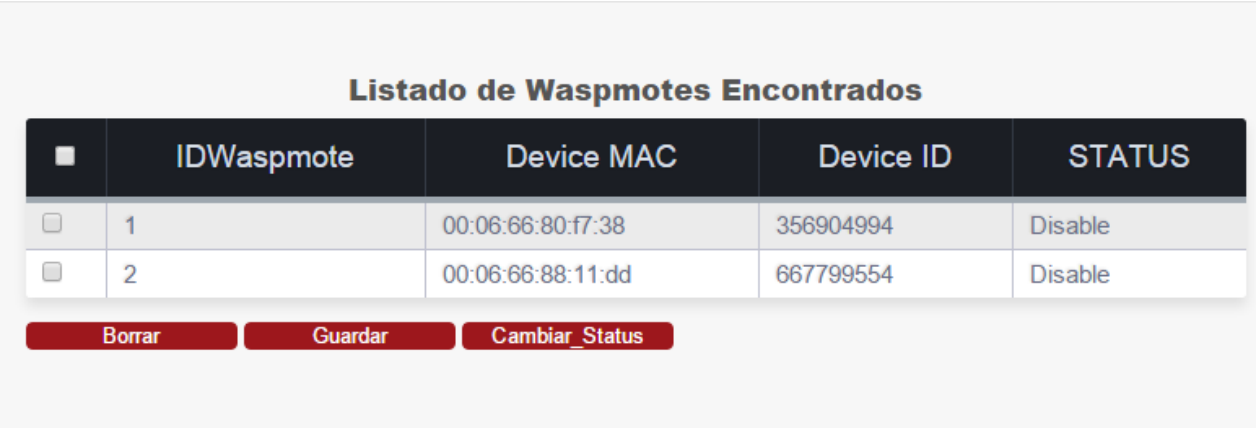

Figura 67. Lista de dispositivos deshabilitados.

Al tratar de enviar lecturas al servidor con el campo STATUS deshabilitado, el servidor responderá con la cadena “Error|Waspmote no habilitada en el entorno”. La Figura 68 es una captura de la interfaz “Serial Monitor” que muestra la respuesta enviada por el sistema al enviar datos con una placa sin habilitar o no existente en el sistema. Por supuesto, estos datos no son guardados en la base de datos del sistema.

```
set i h 192.168.1.34
set i r 80
set c r GET$/W?t=22_09_15_21_06_44&u=FE05ECEB&i=356904994&m=00066680f738
set o f 1
set u m 0
open
open
<2.32> *OPEN*
R: HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 42
Date: Tue, 22 Sep 2015 19:06:53 GMT
Connection: close

Error|Waspmote no habilitada en el entorno*CLOS*
enter CMD at 115200

----> HTTP query OK.
WIFI.answer:
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 42
Date: Tue, 22 Sep 2015 19:06:53 GMT
Connection: close

Error|Waspmote no habilitada en el entorno*CLOS*
----> No encontramos la marca de respuesta positiva en WIFI.answer
--------> Almacenamos la request en la SD
*OFF
**************************
```

Figura 68. Captura de una respuesta del servidor al enviar con una placa no habilitada.

6.2 Pruebas de la interfaz Web

Las pruebas relativas a la funcionalidad de la interfaz web tienen como objetivo probar la funcionalidad de los casos más importantes para el desarrollo del proyecto. Es necesario señalar que todas las pruebas se han realizado sobre el navegador web “Mozilla Firefox”.

Para la probar la funcionalidad de la aplicación web se utilizara unos casos de prueba con el siguiente formato.

Nombre prueba	Descripción	Contexto	Resultado
Buscar admin	Realizar una búsqueda de administradores por cualquiera de los parámetros del registro	Búsqueda realizada con administrador “FULL ADMIN”.	OK
Habilitar/desactivar Admin	Habilitar/desactivar usuario administrador	Habilitar usuario desde una búsqueda.	OK

Editar Admin	Editar campos de usuario administrador. Incluso modificar su rol.	Editar usuario desde una búsqueda	OK
Eliminar Admin	Eliminar registro de usuario administrador.	Eliminar usuario desde una búsqueda. Marcando un checkbox de la tabla.	OK
Crear Admin Mail	Crear nuevo usuario administrador	Crear usuario con envío de correo de activación	OK
Crear Admin	Crear nuevo usuario administrador	Crear usuario sin correo de activación	OK
Buscar Etiqueta	Realizar una búsqueda de etiquetas por cualquiera de los parámetros del registro	Búsqueda realizada con administrador "FULL ADMIN".	OK
Buscar Producto	Realizar una búsqueda de Producto por cualquiera de los parámetros del registro	Búsqueda realizada con administrador "FULL ADMIN".	OK
Buscar Zona	Realizar una búsqueda de Zona por cualquiera de los parámetros del registro	Búsqueda realizada con administrador "FULL ADMIN".	OK
Buscar Waspmote	Realizar una búsqueda de Waspmote por cualquiera de los parámetros del registro	Búsqueda realizada con administrador "FULL ADMIN".	OK
Buscar Responsable	Realizar una búsqueda de Responsables por cualquiera de los parámetros del registro	Búsqueda realizada con administrador "FULL ADMIN	OK
Eliminar Etiquetas	Eliminar registro de Etiqueta.	Eliminar Etiqueta desde una búsqueda. Marcando un checkbox de la tabla.	OK
Eliminar Producto	Eliminar registro de Producto.	Eliminar Producto desde una búsqueda. Marcando un checkbox de la tabla.	OK
Eliminar Zona	Eliminar registro de Zona.	Eliminar Zona desde una búsqueda. Marcando un checkbox de la tabla.	OK
Eliminar Waspmote	Eliminar registro de Waspmote	Eliminar Waspmote desde una búsqueda. Marcando un checkbox de la tabla.	OK
Eliminar Responsable	Eliminar registro de Responsable	Eliminar Responsable desde una búsqueda. Marcando un checkbox de la tabla.	OK
Editar Producto	Editar registro de Producto	Editar Producto desde una búsqueda	OK
Editar Zona	Editar registro de Zona	Editar Zona desde una búsqueda	OK
Editar Waspmote	Editar registro de Waspmote	Editar Waspmote desde una búsqueda	OK
Editar Responsable	Editar registro de Responsable	Editar Responsable desde una búsqueda	OK

Habilitar/desactivar Waspmote	Habilitar/desactivar dispositivo Waspmote para envió de lecturas.	Habilitar Waspmote desde una búsqueda.	OK
Listar Históricos	Listar históricos desde cualquier parámetro.	Listado realizado con administrador "FULL ADMIN".	OK

Tabla 16. Pruebas realizada en la funcionalidad de la interfaz WEB.

6.3 Retardo

Aunque el rendimiento no se ha evaluado sistemáticamente, dado que no era un aspecto crítico para los objetivos del proyecto, sí que se ha observado un retardo significativo en el envío de datos al servidor. A modo de ejemplo se indica en la Figura 69 un caso concreto en el que el retardo es 27 segundos desde que un dato es leído hasta que es enviado. Este retardo es representativo del que podemos observar normalmente en el uso de la aplicación.

```
--------> La hora de la lectura es:[Day of week, YY/MM/DD, hh:mm:ss]Sun, 15/10/18, 20:55:36
----> Parpadeo LED Verde
----> Encendido LED Rojo [Equipo ocupado]

----> Encendemos modulo WIFI
*OFF
enter CMD at 115200
----> Asociacion exitosa con la WLAN
----> Preparamos variables para la Request HTTP
--------> Variable 1: UID de tarjeta detectadaFE05ECEB
--------> Variable 2: Serial del dispositivo: 356904994
--------> Variable 3: MAC del dispositivo [formato sin puntos]00:06:66:80:f7:38
00066680f738

----> Encendido LED Rojo [Equipo ocupado]
--------> Variable 3: Preparamos la marca de tiempo
--------> La fecha actual es:[Day of week, YY/MM/DD, hh:mm:ss]Sun, 15/10/18, 20:55:48
----> El formato de la cadena Request con marca de tiempo es:GET$/W?t=18_10_15_20_55_48&u=FE05ECEB&i=356904994&m=00066680f738
set i h 192.168.1.34
set i r 80
set c r GET$/W?t=18_10_15_20_55_48&u=FE05ECEB&i=356904994&m=00066680f738
set o f 1
set u m 0
open
open
<2.32> *OPEN*
R: HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Content-Length: 7
Date: Sun, 18 Oct 2015 18:55:59 GMT
Connection: close

OK|true*CLOS*
enter CMD at 115200

----> HTTP query OK.
WIFI.answer:
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charset=UTF-8
Content-Length: 7
Date: Sun, 18 Oct 2015 18:55:59 GMT
Connection: close

OK|true*CLOS*
----> Encontramos la marca de respuesta positiva en WIFI.answer
--------> La hora tras el envio:[Day of week, YY/MM/DD, hh:mm:ss]Sun, 15/10/18, 20:56:03
*OFF
**************************************
```

Figura 69. Tiempo de envío de una lectura RFID

Creemos que este retardo representa un problema potencial para el usuario final, puesto que parece lo suficientemente grande como para terminar consumiendo un tiempo bastante considerable en la realización de un inventario completo.

Además, dado el carácter exploratorio del proyecto nos ha parecido importante destacarlo pues creemos que puede ser un problema para el uso de esta configuración de Waspmote en otras aplicaciones.

6.4 Conclusiones del capítulo

A pesar del retardo descrito en el punto 6.3 los resultados de las pruebas realizadas en este capítulo han permitido validar la funcionalidad del sistema de control de inventario desarrollado.

Capítulo 7

Conclusiones y líneas de trabajo futuras

7.1 Conclusiones

Sin duda las WSNs son una tecnología de futuro por su gran versatilidad y su cada vez más asumible coste. Junto con RFID sus posibilidades en campos como el militar, la salud o simplemente las nuevas posibilidades que puedan llegar con la "Internet of Things", se ven ampliadas.

En cuanto al campo de aplicación de este trabajo, la gestión de almacenes y automatización de inventario, existen proyectos de investigación, como ya vimos en el punto 2.2.3, en los que estas tecnologías son fuente de estudio por los beneficios de su aplicación. Las ventajas más importantes que nos proporcionan son la automatización, reduciendo al mínimo la interacción humana con el fin de minimizar errores; y el control de materias primas peligrosas, para facilitar su control y su inventariado evitando accidentes para el ser humano o el medio ambiente.

En lo relativo a nuestro trabajo podemos concluir que se han conseguido los objetivos marcados inicialmente: obtener experiencia en el desarrollo con Waspmote y plasmarlo en un proyecto. Pero, aunque el módulo WiFi aporta la ventaja de la interoperabilidad con las redes inalámbricas 802.11, redes que han tenido mucha penetración en el mercado, no parece ser un terreno suficientemente maduro en la plataforma Waspmote. A lo largo del desarrollo nos hemos encontrado multitud de problemas con la librería WiFi en Waspmote y, aunque estos problemas están solventados en las últimas versiones de la API, el rendimiento del módulo WiFi no parece ser suficientemente rápido en el desempeño necesario para un dispositivo que tiene la tarea marcada en el presente proyecto, actuar como un lector RFID para la realización de inventarios.

En cuanto al protocolo escogido a la hora de reportar la información al servidor, HTTP, el dispositivo se muestra lento en su desempeño a la hora de establecer sesiones, algo que termina influyendo en el rendimiento de toda la plataforma.

Por todo lo expuesto sobre la integración de Waspmote en este proyecto, podemos concluir que, si bien la herramienta es totalmente funcional, tal vez no sea la herramienta óptima para el diseño llevado a cabo. Creemos que Waspmote es profundamente versátil y nos hubiera gustado tener la posibilidad de probar otras configuraciones de la plataforma para determinar si el problema de rendimiento podría haberse evitado si en lugar de utilizar el módulo WiFi como módulo de comunicación y HTTP como protocolo de envió de datos, se hubiera utilizado cualquier otra opción y ver si así se obtendrían resultados más satisfactorios. A continuación, en el punto 7.2 se exponen algunas configuraciones e ideas alternativas que podrían ayudar a encontrar un mejor diseño para un sistema de control de inventario.

7.2 Líneas de trabajo futuras

Las líneas de trabajo futuras podrían dividirse en:

- **Configuraciones alternativas de la plataforma Waspmote:** Una de las líneas de trabajo futuras sería investigar y probar nuevas configuraciones de la plataforma diferentes de las descritas en este documento.

  Una opción es el uso de topologías jerárquicas. Como se ha comentado con anterioridad, el uso de topologías jerárquicas con los módulos WiFi requiere de desarrollo al margen de la API.

  Pero este punto queremos enfocarlo de otra manera. Una solución interesante sería una topología jerárquica en la que los nodos lectores monten módulos ZigBee como módulo de comunicaciones y los nodos Cluster Head monten un módulo ZigBee y un módulo WiFi para enrutar los paquetes del resto de nodos a la WLAN. Esto permitiría:

  - Mantener la interoperabilidad con redes WiFi por medio del Cluster Head.
  - Continuar enviando los paquetes utilizando el protocolo HTTP.
  - Y, creemos, que podría mejorar el rendimiento de los nodos lectores.

- **Mejoras en el diseño actual:** El diseño actual tiene muchas posibilidades de ser ampliado, tanto por características que se han quedado sin investigar como por características que se han quedado fuera del alcance del proyecto:

- Envío de datos desde la SD: Esta funcionalidad sea tal vez, la más fácil de implementar. El desarrollo actual ya almacena los envíos HTTP que han tenido algún problema en el envío en un archivo de texto en la tarjeta SD. Por lo que solo habría que leerlos del fichero y enviarlos cuando la placa no esté realizando ninguna otra tarea.

- HTTPS: De nuevo, esta opción no está incluida en la librería WiFi. Por lo que habría que implementar una capa SSL/TLS sobre la que realizar la solicitud HTTP.

- Interrupciones con el módulo RFID: Aunque parece que el diseño del dispositivo RFID no está pensado para trabajar con interrupciones, sería optimo disponer de la posibilidad de, durante un inventario, si el dispositivo tiene un tiempo de inactividad prolongado poder entrar en un modo de consumo reducido y poder devolverlo al modo ON por medio de una interrupción de lectura del RFID.

- Pantalla LCD: La inclusión de una pantalla LCD en el dispositivo permitiría mejorar la interacción de los usuarios con el hardware. Además acercaría el diseño a las funcionalidades de un dispositivo comercial.

- **Mejoras en la aplicación Web:** A continuación se enumeran algunas ideas sobre algunos puntos que podrían incurrir en mejoras interesantes para la aplicación web:

  - Interfaz visual del entorno a inventariar: Incluir en el menú una opción que permita al administrador obtener una visual del edificio o área a inventariar en la que distinguir las diferentes zonas del sistema con los distintos ítems en tiempo real.

  - Posibilidad de exportar informes: La posibilidad de realizar informes y exportarlos en ficheros como PDF o CSV.

  - Configuración de HTTPS en la interfaz web: Configurar en el sistema el servicio HTTPS para la administración web.

Capítulo 8

Planificación y presupuesto

El siguiente capítulo describe de forma gráfica las fases en la que se ha divido el conjunto del proyecto. Se incluyen también todos los aspectos concernientes al presupuesto del proyecto.

8.1 Planificación

Para facilitar el desarrollo y obtener una compresión global del mismo se ha divido el proyecto en distintas fases o hitos. Los diagramas de Gantt nos permiten mostrar esta información y además nos proporción una representación cronológica del tiempo acumulado en el desarrollo.

A continuación se incluyen los diagramas de Gantt que aplican a las tareas del proyecto. La Figura 70 contiene las distintas fases y tareas que componen el proyecto así como su duración. La Figura 71 incluye las distintas fases en un gráfico de barras que muestra las distintas tareas que han podido realizarse en paralelo.

Nombre	Fecha de inicio	Fecha de fin	Duración
Automatización del control de inventario mediante redes de sensores	6/02/13	30/09/15	691
FASE-1 Analisis e investigación	6/02/13	31/12/13	235
Estudio de las WSN	6/02/13	31/05/13	83
Aplicaciones de control de inventario con redes de sensores	1/04/13	31/07/13	88
Análisis de las posibilidades Waspmote	1/08/13	31/12/13	109
FASE-2 Diseño	1/01/14	30/06/14	129
Definición de requisitos	1/01/14	28/02/14	43
Diseño de la Base de datos	28/02/14	18/03/14	13
Diseño de la lógica de la apliación	19/03/14	9/06/14	59
Definición de los servicios adicionales	21/04/14	9/06/14	36
Diseño objetivo de la vista de la aplicación WEB	10/06/14	30/06/14	15
FASE-3 Desarrollo	1/07/14	10/08/15	290
Despliegue del entorno de desarrollo	1/07/14	18/07/14	14
Instalación del OS (Ubuntu 12.04LTS)	1/07/14	1/07/14	1
Instalación del IDE para Waspmote	1/07/14	3/07/14	3
Instalación de Java	3/07/14	3/07/14	1
Instalación de MySQL	3/07/14	3/07/14	1
Instalación del servicio NTP	3/07/14	3/07/14	1
Instalación y configuración de Bind	4/07/14	11/07/14	6
Instalación y configuración de Postscript	11/07/14	18/07/14	6
Scripts de la BBDD	18/07/14	1/09/14	32
Script de creación de tablas	18/07/14	31/07/14	10
Script de datos de prueba	31/07/14	1/09/14	23
Implementación	2/09/14	10/08/15	245
Código de Waspmote	2/09/14	30/01/15	109
Código del sistema de Control de inventario	10/11/14	10/08/15	196
Librerías	1/12/14	16/03/15	76
Libreria de envío de emails	1/12/14	16/02/15	56
Libreria de encriptación de contraseñas	1/01/15	16/03/15	53
Clases JavaBeans	17/03/15	30/03/15	10
Clases para las interfases de BDD	17/03/15	30/04/15	33
Servlets y JSP	10/11/14	10/08/15	196
Entrada de datos Waspmote	10/11/14	30/01/15	60
Entrada y salida de datos	1/05/15	10/08/15	72
Autenticación de usuarios	15/06/15	10/08/15	41
FASE-4 Pruebas	10/08/15	21/08/15	10
Pruebas de Waspmote	10/08/15	13/08/15	4
Pruebas de sincronización NTP de Waspmote	10/08/15	12/08/15	3
Pruebas de envío de datos HTTP	11/08/15	13/08/15	3
Pruebas en envío de datos con el status deshabilitado	13/08/15	13/08/15	1
Pruebas de la interfaz Web	14/08/15	21/08/15	6

Figura 70. Diagrama con la planificación de las distintas fases del proyecto

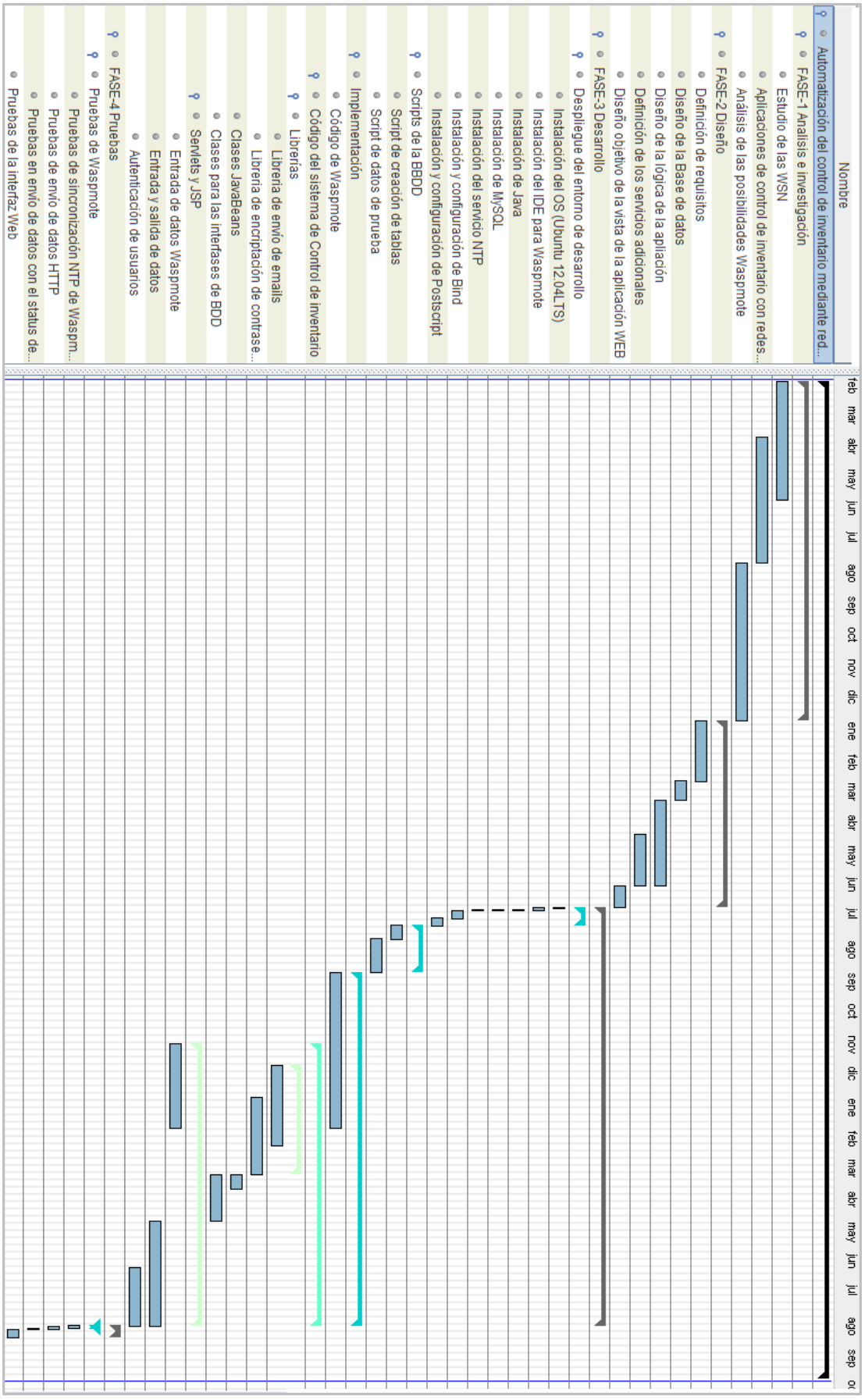

Figura 71. Diagrama de Gantt

8.2 Presupuesto

En capítulo se muestra el presupuesto total para el desarrollo del sistema de control de inventario. El detalle de los gastos del proyecto se encuentra dividido en gastos de personal, gasto de equipos, subcontratación de tareas y otros costes directos. El presupuesto completo se encuentra en la Figura 72.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor: **Pablo Romaus Campos**

2.- Departamento: **Telemática**

3.- Descripción del Proyecto:

- Titulo: **Automatización del control del inventario mediante redes de sensores**
- Duración (meses): **30**
- Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):

Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F.	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Soto Campos, Ignacio		Ingeniero Senior	1	4.289,54	4.289,54	
Romaus Campos, Pablo		Ingeniero	6	2.694,39	16.166,34	
		Hombres mes 7		**Total**	20.455,88	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
PC Laboratorio	200,00	100	12	60	40,00
Sobremesa Packard Bell iMax	500,00	10	12	60	10,00
Portatil Asus F555L	650,00	90	6	60	58,50
2xWaspmoteProV1,2 + 2xMódulos WiFi	300,00	100	22	60	110,00
2xMódulos RFID	100,00	100	22	60	36,67
2xTarjetasRFID+4xKeyring RFID	6,00	100	22	60	2,20
Router wifi linksys wrt54g	59,00	20	22	60	4,33
				Total	261,69

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} \text{xCxD}$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO [e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	20.456
Amortización	262
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	4.144
Total	24.861

Figura 72. Presupuesto completo del proyecto

Capítulo 9

Anexos

9.1 Anexo I. Requisitos

Se ha utilizado el siguiente formato de campos para definir los distintos requisitos:

- Identificador: Referencia de manera unívoca el requisito. El formato del nombre del identificador seguirá la estructura:
    - RU-XX: La letras RU identifican a los requisitos de usuario.
    - RSF-XX: La letras RSF hacen referencia a los requisitos funcionales de software
    - RSNF-XX: La letras RSNF hacen referencia a los requisitos no funcionales de software.

  Siendo XX tras las letras un número comprendido entre 01 y 99.
- Nombre: Nombre del requisito.
- Fuente: Referencia el origen de este requisito.
- Descripción: Breve explicación que explica en qué consiste el requisito.

A. Requisitos de Usuario

Los requisitos de usuario abarcan las demandas realizadas por el usuario final de la aplicación y que tienen como objetivo definir los puntos que debe de cumplir la aplicación.

Identificador: RU-01	
Nombre:	Interfaz web
Descripción:	La aplicación contará con una consola de administración web.
Fuente:	Tutor

Tabla 17. RU-01. Interfaz web

Identificador: RU-02	
Nombre:	Roles de administración.
Descripción:	La aplicación deberá de contar con 3 roles de administración: • FULL ADMIN • BDD ADMIN • REPORT ADMIN
Fuente:	Tutor

Tabla 18. RU-02. Roles de administración

Identificador: RU-03	
Nombre:	Crear usuarios administradores.
Descripción:	Los administradores con el rol FULL ADMIN podrán crear nuevos administradores.
Fuente:	Desarrollador

Tabla 19. RU-03. Crear usuarios administradores

Identificador: RU-04	
Nombre:	Crear usuarios administradores activables por mail con una password.
Descripción:	Los administradores con el rol FULL ADMIN al crear nuevos administradores podrán asignarles una password directamente y enviársela al administrador por email. Este usuario se encontrara inhabilitado hasta que se active.
Fuente:	Desarrollador

Tabla 20. RU-04. Crear usuarios administradores activables por mail con una password

Identificador: RU-05	
Nombre:	Crear usuarios administradores activos con una password.
Descripción:	Los administradores con el rol FULL ADMIN al crear nuevos administradores podrán asignarles una password directamente y no enviársela por email. Este usuario se encontrará habilitado.
Fuente:	Desarrollador

Tabla 21. RU-05. Crear usuarios administradores activos con una password

Identificador: RU-06	
Nombre:	Búsquedas de usuarios administradores.
Descripción:	Los administradores con el rol FULL ADMIN podrán realizar búsquedas de los administradores del sistema.
Fuente:	Desarrollador

Tabla 22. RU-06. Búsquedas de usuarios administradores

Identificador: RU-07	
Nombre:	Borrar usuarios administradores.
Descripción:	Los administradores con el rol FULL ADMIN podrán borrar administradores del sistema.
Fuente:	Desarrollador

Tabla 23. RU-07. Borrar usuarios administradores

Identificador: RUC-08	
Nombre:	Inhabilitar usuarios administradores.
Descripción:	Los administradores con el rol FULL ADMIN podrán inhabilitar administradores del sistema.
Fuente:	Desarrollador

Tabla 24. RU-08. Inhabilitar usuarios administradores

Identificador: RUC-09	
Nombre:	Habilitar usuarios administradores.
Descripción:	Los administradores con el rol FULL ADMIN podrán inhabilitar administradores del sistema.
Fuente:	Desarrollador

Tabla 25. RU-09. Habilitar usuarios administradores

Identificador: RU-10	
Nombre:	Modificar el rol de usuarios administradores.
Descripción:	Los administradores con el rol FULL ADMIN podrán modificar el rol de otros usuarios administradores.
Fuente:	Desarrollador

Tabla 26. RU-10. Modificar el rol de usuarios administradores

Identificador: RU-11	
Nombre:	Insertar datos en la base de datos.
Descripción:	Los administradores con el rol FULL ADMIN y BDD ADMIN podrán insertar datos relativos a la lógica de negocio en la base de datos.
Fuente:	Tutor

Tabla 27. RU-11. Insertar datos en la base de datos

Identificador: RU-12	
Nombre:	Eliminar datos en la base de datos.
Descripción:	Los administradores con el rol FULL ADMIN y BDD ADMIN podrán eliminar datos relativos a la lógica de negocio en la base de datos.
Fuente:	Tutor

Tabla 28. RU-12. Eliminar datos en la base de datos

Identificador: RU-13	
Nombre:	Modificar datos en la base de datos
Descripción:	Los administradores con el rol FULL ADMIN y BDD ADMIN podrán modificar datos relativos a la lógica de negocio en la base de datos.
Fuente:	Tutor

Tabla 29. RU-13. Modificar datos en la base de datos

Identificador: RU-14	
Nombre:	Guardar históricos de inventario
Descripción:	La aplicación permitirá almacenar históricos de los inventarios.
Fuente:	Tutor

Tabla 30. RU-14. Guardar históricos de inventario

Identificador: RU-15	
Nombre:	Listar históricos de inventario
Descripción:	La aplicación permitirá listar históricos de los inventarios a todos los usuarios administradores.
Fuente:	Tutor

Tabla 31. RU-15. Listar históricos de inventario

Identificador: RU-16	
Nombre:	No eliminar históricos
Descripción:	Ningún usuario administrador podrá eliminar históricos.
Fuente:	Desarrollador

Tabla 32. RC-16. No eliminar históricos.

Identificador: RU-17	
Nombre:	No modificar históricos
Descripción:	Ningún usuario administrador podrá modificar los históricos.
Fuente:	Desarrollador

Tabla 33. RU-17. No modificar históricos.

Identificador: RU-18	
Nombre:	Realizar búsquedas en la base de datos.
Descripción:	Todos los administradores podrán realizar búsquedas de datos relativos a la lógica de negocio en la base de datos.
Fuente:	Tutor

Tabla 34. RU-18. Realizar búsquedas en la base de datos

Identificador: RU-19	
Nombre:	Las Waspmote enviaran los datos al servidor
Descripción:	Las Waspmote deben enviar los datos leídos con el módulo RFID al servidor.
Fuente:	Tutor

Tabla 35. RU-19. Las Waspmote enviaran los datos al servidor.

Identificador: RU-20	
Nombre:	Las Waspmote usaran HTTP
Descripción:	Las Waspmote deben enviar los datos utilizando el protocolo HTTP.
Fuente:	Tutor

Tabla 36. RU-20. Las Waspmote usaran HTTP.

Identificador: RU-21	
Nombre:	Inhabilitar Waspmote en la base de datos.
Descripción:	Todos los administradores podrán inhabilitar dispositivos Waspmote para él envió de datos al servidor.
Fuente:	Desarrollador

Tabla 37. RU-21. Inhabilitar Waspmote en la base de datos

Identificador: RU-22	
Nombre:	Sincronización NTP
Descripción:	Los dispositivos Waspmote deberán sincronizarse por NTP con el servidor del entorno.
Fuente:	Desarrollador

Tabla 38. RU-22. Sincronización NTP

Identificador: RU-23	
Nombre:	Enviar correos de alerta.
Descripción:	El sistema debe enviar un correo de alerta al responsable de un ítem cuando se produzca el cambio de ubicación del producto.
Fuente:	Tutor

Tabla 39. RU-23. Enviar correos de alerta

Identificador: RU-24	
Nombre:	Encriptación de contraseñas
Descripción:	Las contraseñas de los administradores deben estar encriptadas en la base de datos.
Fuente:	Tutor

Tabla 40. RU-24. Encriptación de contraseñas

B. Requisitos de Software

a. Requisitos funcionales

Los requisitos funcionales son una declaración de los servicios que proporciona el sistema.

Identificador: RSF-01	
Nombre:	Implementar gestión de usuarios administradores
Descripción:	Implementar una solución que permita, mediante la interfaz web, realizar las siguientes operaciones: • Crear usuarios administradores. • Borrar usuarios administradores. • Habilitar e desactivar usuarios administradores. • Realizar búsquedas de administradores. • Modificar el Rol de otros administradores.
Fuente:	RU-01/RU-02/RU-03/RU-06/RU-07/RU-08/RU-09/RU-10

Tabla 41. RSF-01. Implementar gestión de usuarios administradores

Identificador: RSF-02	
Nombre:	Implementar gestión de la base de datos de negocio
Descripción:	Implementar una solución que permita, mediante la interfaz web, realizar las siguientes operaciones: • Insertar datos de la base de datos de negocio. • Borrar datos de la base de datos de negocio. • Realizar búsquedas sobre la base de datos de negocio. Y no permita: • Eliminar históricos de la base de datos de negocio. • Modificar históricos de la base de datos de negocio.
Fuente:	RU-01/RU-11/RU-12/RU-13/RU-14/RU-17/RU-18

Tabla 42. RSF-02. Implementar gestión de la base de datos de negocio

Identificador: RSF-03	
Nombre:	Implementar una interfaz con la base de datos de administradores
Descripción:	Implementar una clase que actué de interfaz con la BDD. Y que disponga de los métodos necesarios para realizar operaciones de inserción, borrado y modificación de registros sobre la base de datos de administradores
Fuente:	RU-03/ RU-06/RU-07/RU-08/RU-09/RU-10/Desarrollador

Tabla 43. RSF-03. Implementar una interfaz con la base de datos de administradores

Identificador: RSF-04	
Nombre:	Implementar una interfaz con la base de datos de negocio
Descripción:	Implementar una clase que actué de interfaz con la BDD. Y que disponga de los métodos necesarios para realizar operaciones de inserción, borrado y modificación de registros sobre la base de datos de la lógica de negocio
Fuente:	RU-11/ RU-12/RU-13/RU-14/RU-15/RU-18/RU-21/Desarrollador

Tabla 44. RSF-04. Implementar una interfaz con la base de datos de negocio

Identificador: RSF-05	
Nombre:	Servicio HTTP para Waspmote
Descripción:	El sistema debe contar con una solución HTTP para establecer la comunicación con los dispositivos Waspmote.
Fuente:	RC-20

Tabla 45. RSF-05. Servicio HTTP para Waspmote

Identificador: RSF-06	
Nombre:	Sincronización NTP
Descripción:	El sistema de la plataforma debe contar con una aplicación que permita la sincronización por NTP con los dispositivos Waspmote. Esta aplicación formara parte del sistema operativo del equipo servidor pero no de la solución a desarrollar.
Fuente:	RU-22/Desarrollador

Tabla 46. RSF-06. Sincronización NTP

Identificador: RSF-07	
Nombre:	Enviar correos
Descripción:	La aplicación debe contar con un módulo que permita enviar correos electrónicos. Esta aplicación enviara los correos consultando los registros MX del dominio de destino.
Fuente:	RU-04/RU-23

Tabla 47. RSF-07. Enviar correos

Identificador: RSF-08	
Nombre:	Encriptar contraseñas
Descripción:	La aplicación debe contar con un módulo de encriptación de contraseñas. Las contraseñas se deben almacenar encriptadas en la base de datos de administradores. El método de encriptación no permitirá desencriptar las contraseñas almacenadas.
Fuente:	RU-24

Tabla 48. RSF-08. Encriptar contraseñas

b. Requisitos funcionales

Los requisitos no funcionales son aquellos que no están directamente relacionados con funciones que realiza el software.

Identificador: RSNF-01	
Nombre:	Interfaz Web
Descripción:	El sistema debe contar con una interfaz Web. Esta interfaz estará basada en Servlets, JSPs, AJAX, JavaScript, HTML5 y CSS3.
Fuente:	RU-1/ Tutor

Tabla 49. RSNF-01. Interfaz Web

Identificador: RSNF-02	
Nombre:	Base de datos de negocio
Descripción:	El sistema debe contar con una base de datos MySQL para el almacenamiento de datos relativos a lógica de negocio.
Fuente:	Desarrollador

Tabla 50. RSNF-02. Base de datos de negocio

Identificador: RSNF-03	
Nombre:	Base de datos de Administradores
Descripción:	El sistema debe contar con una base de datos MySQL para el almacenamiento de datos de la información de administradores.
Fuente:	Desarrollador

Tabla 51. RSNF-03. Base de datos de Administradores

Identificador: RSNF-04	
Nombre:	Tabla de un usuario administrador
Descripción:	El registro de un administrador de la tabla administradores debe contar con los siguientes campos: • Nombre • Rol • Email • Status
Fuente:	RU-02/ RU-04

Tabla 52. RSNF-04. Tabla de un usuario administrador

Identificador: RSNF-05	
Nombre:	URL de activación
Descripción:	El correo electrónico para activación de usuarios dispondrá de una URL donde el destinatario podrá activar el usuario administrador.
Fuente:	RU-04

Tabla 53. RSNF-05. URL de activación

Identificador: RSNF-06	
Nombre:	Sistema operativo del entorno desarrollo
Descripción:	El sistema operativo del entorno de desarrollo será Ubuntu 12.04LTS.
Fuente:	Tutor

Tabla 54. RSNF-06. Sistema operativo del entorno desarrollo

Identificador: RSNF-08	
Nombre:	Desarrollo con software libre
Descripción:	El entorno de desarrollo deberá realizarse con software libre siempre que sea posible.
Fuente:	Tutor

Tabla 55. RSNF-08. Desarrollo con software libre

9.2 Anexo II. Registros DNS

A. Registros para resolución directa

```
;
; BIND data file for local loopback interface
;
$TTL  604800
@     IN    SOA   ns1.findbackend.local. admin.findbackend.local. (
                        3           ; Serial
                   604800           ; Refresh
                    86400           ; Retry
                  2419200           ; Expire
                   604800 )   ; Negative Cache TTL
;
;NS-Records
      IN    NS      ns1.findbackend.local.
;MX-Record
      IN    MX    10 mail.findbackend.local.

ns1.findbackend.local.  IN    A     192.168.1.34
mail.findbackend.local. IN    A     192.168.1.34
ntp.findbackend.local.  IN    A     192.168.1.34
findbackend.local.      IN    A     192.168.1.34
www.findbackend.local.  IN    A     192.168.1.34
```

B. Registros para resolución inversa

```
;
;BIND reverse data file for local loopback interface
;
$TTL  604800
@     IN    SOA   ns1.findbackend.local. admin.findbackend.local. (
                        3           ; Serial
                   604800           ; Refresh
                    86400           ; Retry
                  2419200           ; Expire
                   604800 )   ; Negative Cache TTL
;
;NS-Records
      IN    NS      ns1.findbackend.local.

;PTR-Records
34    IN    PTR   ns1.findbackend.local.
34    IN    PTR   mail.findbackend.local.
34    IN    PTR   ntp.findbackend.local.
34    IN    PTR   findbackend.local.
34    IN    PTR   www.findbackend.local.
```

9.3 Anexo III. Scripts de la Base de datos

A. Scripts de la base de datos Negocio.

a. Script de creación.

```
DROP TABLE if exists  Negocio.EtiquetasZona, Negocio.EtiquetasItem,
Negocio.Etiquetas, Negocio.Waspmote, Negocio.Items, Negocio.Zonas,
Negocio.Responsable, Negocio.Historic;

DROP DATABASE if exists Negocio;

CREATE DATABASE Negocio;

CREATE TABLE Negocio.Responsable (
IdResponsable INT NOT NULL auto_increment,
Nombre VARCHAR(50) NOT NULL,
Apellidos VARCHAR(100) NOT NULL,
Cargo VARCHAR(100),
Email VARCHAR (150) NOT NULL,
PRIMARY KEY(IdResponsable),
UNIQUE INDEX (Email)
) ENGINE = INNODB;

CREATE TABLE Negocio.Zonas (
IdZona INT NOT NULL auto_increment,
IdResponsable INT NOT NULL,
Edificio VARCHAR(50) NOT NULL,
Planta VARCHAR(15) NOT NULL,
Letra VARCHAR(15) NOT NULL,
Sala VARCHAR(15) NOT NULL,
Departamento TEXT NOT NULL,
PRIMARY KEY(IdZona),
INDEX (IdResponsable),
FOREIGN KEY (IdResponsable) REFERENCES
Negocio.Responsable(IdResponsable) ON UPDATE CASCADE
) ENGINE=INNODB;

CREATE TABLE Negocio.Items (
IdItem INT NOT NULL auto_increment,
IdResponsable INT NOT NULL,
NumeroSerie VARCHAR(80) NOT NULL,
FechaAlta DATE,
FechaBaja DATE,
Descripcion TEXT,
PRIMARY KEY (IdItem),
UNIQUE INDEX (NumeroSerie),
INDEX (IdResponsable),
FOREIGN KEY (IdResponsable) REFERENCES
Negocio.Responsable(IdResponsable) ON UPDATE CASCADE
) ENGINE = INNODB;

CREATE TABLE Negocio.Etiquetas(
UID INT UNSIGNED NOT NULL,
Tipo VARCHAR(1) NOT NULL,
PRIMARY KEY (UID)
) ENGINE =INNODB;
```

```
CREATE TABLE Negocio.EtiquetasZona(
UID  INT UNSIGNED NOT NULL,
IdZona INT NOT NULL,
PRIMARY KEY (UID),
INDEX(IdZona),
FOREIGN KEY (UID) REFERENCES Negocio.Etiquetas(UID) ON UPDATE CASCADE,
FOREIGN KEY (IdZona) REFERENCES Negocio.Zonas(IdZona) ON UPDATE CASCADE
) ENGINE =INNODB;

CREATE TABLE Negocio.EtiquetasItem(
UID INT UNSIGNED NOT NULL,
IdZona INT NOT NULL,
IdItem INT NOT NULL,
PRIMARY KEY (UID),
INDEX (IdItem),
FOREIGN KEY (UID) REFERENCES Negocio.Etiquetas(UID) ON UPDATE CASCADE,
FOREIGN KEY (IdZona) REFERENCES Negocio.Zonas(IdZona) ON UPDATE CASCADE,
FOREIGN KEY (IdItem) REFERENCES Negocio.Items(IdItem) ON UPDATE CASCADE
) ENGINE =INNODB;

CREATE TABLE Negocio.Waspmote(
IDWaspmote INT NOT NULL auto_increment,
DeviceMAC VARCHAR (17) NOT NULL,
DeviceID VARCHAR (10) NOT NULL,
STATUS BOOLEAN NOT NULL,
PRIMARY KEY (IDWaspmote),
UNIQUE INDEX (DeviceMAC),
UNIQUE INDEX (DeviceID)
) ENGINE =INNODB;

CREATE TABLE Negocio.Historic(
IDHistoric INT NOT NULL auto_increment,
Fecha DATETIME NOT NULL,
DeviceMAC VARCHAR (17) NOT NULL,
DeviceID VARCHAR (10) NOT NULL,
UID INT UNSIGNED NOT NULL,
IdZona INT,
PRIMARY KEY (IDHistoric),
INDEX (Fecha,DeviceMac,DeviceID)
) ENGINE =INNODB
```

b. Script de datos de prueba.

```
INSERT INTO Negocio.Responsable(Nombre, Apellidos,Cargo, Email) VALUES
('Raul','Roman Reledo','Profesor Titular','mail1@findbackend.local');
INSERT INTO Negocio.Responsable(Nombre, Apellidos,Cargo, Email) VALUES
('Pepe','Pepito Pepote','Profesor Titular','mail2@findbackend.local');
INSERT INTO Negocio.Responsable(Nombre, Apellidos,Cargo, Email) VALUES
('Jose','Raton Perez','Profesor Asociado','mail3@findbackend.local');
INSERT INTO Negocio.Responsable(Nombre, Apellidos,Cargo, Email) VALUES
('Marina','Aguas Claras','Ayudante','mail4@findbackend.local');
INSERT INTO Negocio.Responsable(Nombre, Apellidos,Cargo, Email) VALUES
('Curro','Bastante Bastante','Tecnico
Informatico','mail5@findbackend.local');

INSERT INTO Negocio.Zonas(IdResponsable, Edificio, Planta, Letra, Sala,
Departamento) VALUES ('1','2','2','E','5','Telemática');
```

```
INSERT INTO Negocio.Zonas(IdResponsable, Edificio, Planta, Letra, Sala,
Departamento) VALUES ('2','4','1','F','5','Telemática');
INSERT INTO Negocio.Zonas(IdResponsable, Edificio, Planta, Letra, Sala,
Departamento) VALUES ('3','4','2','F','5','Señales');

INSERT INTO Negocio.Items(IdResponsable, NumeroSerie, FechaAlta,
FechaBaja, Descripcion) VALUES ('2','0123456789','1000-01-01','9999-12-
31','PC sobremesa');
INSERT INTO Negocio.Items(IdResponsable, NumeroSerie, FechaAlta,
FechaBaja, Descripcion) VALUES ('2','7894561230','1000-01-01','9999-12-
31','PC sobremesa');
INSERT INTO Negocio.Items(IdResponsable, NumeroSerie, FechaAlta,
FechaBaja, Descripcion) VALUES ('4','8976452310','1000-01-01','9999-12-
31','PC sobremesa');
INSERT INTO Negocio.Items(IdResponsable, NumeroSerie, FechaAlta,
FechaBaja, Descripcion) VALUES ('3','9876543210','1000-01-01','9999-12-
31','PC sobremesa');
INSERT INTO Negocio.Items(IdResponsable, NumeroSerie, FechaAlta,
FechaBaja, Descripcion) VALUES ('1','8956231470','1000-01-01','9999-12-
31','PC sobremesa');
INSERT INTO Negocio.Items(IdResponsable, NumeroSerie, FechaAlta,
FechaBaja, Descripcion) VALUES ('1','0123423456','1000-01-01','9999-12-
31','Teléfono');
INSERT INTO Negocio.Items(IdResponsable, NumeroSerie, FechaAlta,
FechaBaja, Descripcion) VALUES ('5','0123443210','1000-01-01','9999-12-
31','PC sobremesa');
INSERT INTO Negocio.Items(IdResponsable, NumeroSerie, FechaAlta,
FechaBaja, Descripcion) VALUES ('3','4567887654','1000-01-01','9999-12-
31','Impresora');

INSERT INTO Negocio.Etiquetas(UID, Tipo) VALUES (0xFE05ECEB,'0');
INSERT INTO Negocio.Etiquetas(UID, Tipo) VALUES (0x9E9CECEB,'0');
INSERT INTO Negocio.Etiquetas(UID, Tipo) VALUES (0xFD02A177,'1');
INSERT INTO Negocio.Etiquetas(UID, Tipo) VALUES (0x7227925D,'1');

INSERT INTO Negocio.EtiquetasZona(UID, IdZona) VALUES (0xFE05ECEB,'1');
INSERT INTO Negocio.EtiquetasZona(UID, IdZona) VALUES (0x9E9CECEB,'2');

INSERT INTO Negocio.EtiquetasItem(UID, IdZona, IdItem) VALUES
(0xFD02A177,'2','3');
INSERT INTO Negocio.EtiquetasItem(UID, IdZona, IdItem) VALUES
(0x7227925D,'2','7');

INSERT INTO Negocio.Waspmote(DeviceMac, DeviceID, STATUS) VALUES
('00:06:66:80:f7:38','356904994',true);
```

B. Scripts de la base de datos Admin.

a. Script de creación.

```
DROP TABLE if exists Admin.ManagerStatusURL,Admin.Manager;

DROP DATABASE if exists Admin;

CREATE DATABASE Admin;

CREATE TABLE Admin.Manager(
IdManager INT NOT NULL auto_increment,
Alias VARCHAR(100) NOT NULL,
```

```
Salt BLOB NOT NULL,
Hash BLOB NOT NULL,
Name VARCHAR(50),
Surname VARCHAR(100),
Rol VARCHAR(30) NOT NULL,
Email VARCHAR(150) NOT NULL,
Status BOOLEAN NOT NULL,
PRIMARY KEY (IdManager),
UNIQUE INDEX (Alias),
INDEX (Name, Surname),
INDEX (Rol),
UNIQUE INDEX (Email)
) ENGINE=INNODB;

CREATE TABLE Admin.ManagerStatusURL(
IDStatusURL INT NOT NULL auto_increment,
URL VARCHAR(767) NOT NULL,
Fecha DATETIME NOT NULL,
IdManager INT NOT NULL,
PRIMARY KEY(IDStatusURL),
FOREIGN KEY (IdManager) REFERENCES Admin.Manager(IdManager) ON UPDATE
CASCADE ON DELETE CASCADE,
UNIQUE INDEX (URL)
) ENGINE=INNODB;
```

b. Script de datos de prueba.

```
INSERT INTO Admin.Manager (Alias, Salt, Hash, Name, Surname, Rol, Email,
Status)
VALUES('Administrador','1023e0867dc5e4e192c9497bfd77191aef282d783b7e3b45
','7260bd4390b091878622d879742a3c8c2466f7e29830f0fc',
'Administrador','FULL', 'FULL
ADMIN','admin_local1@findbackend.local',true);
```

9.4 Anexo IV. Manual de usuario

A. Acceso a la consola web.

Para acceder a la interfaz web es necesario contar con una cuenta de administración. Si disponemos de ella simplemente es necesario introducir las credenciales en la pantalla de login (ver Figura 73).

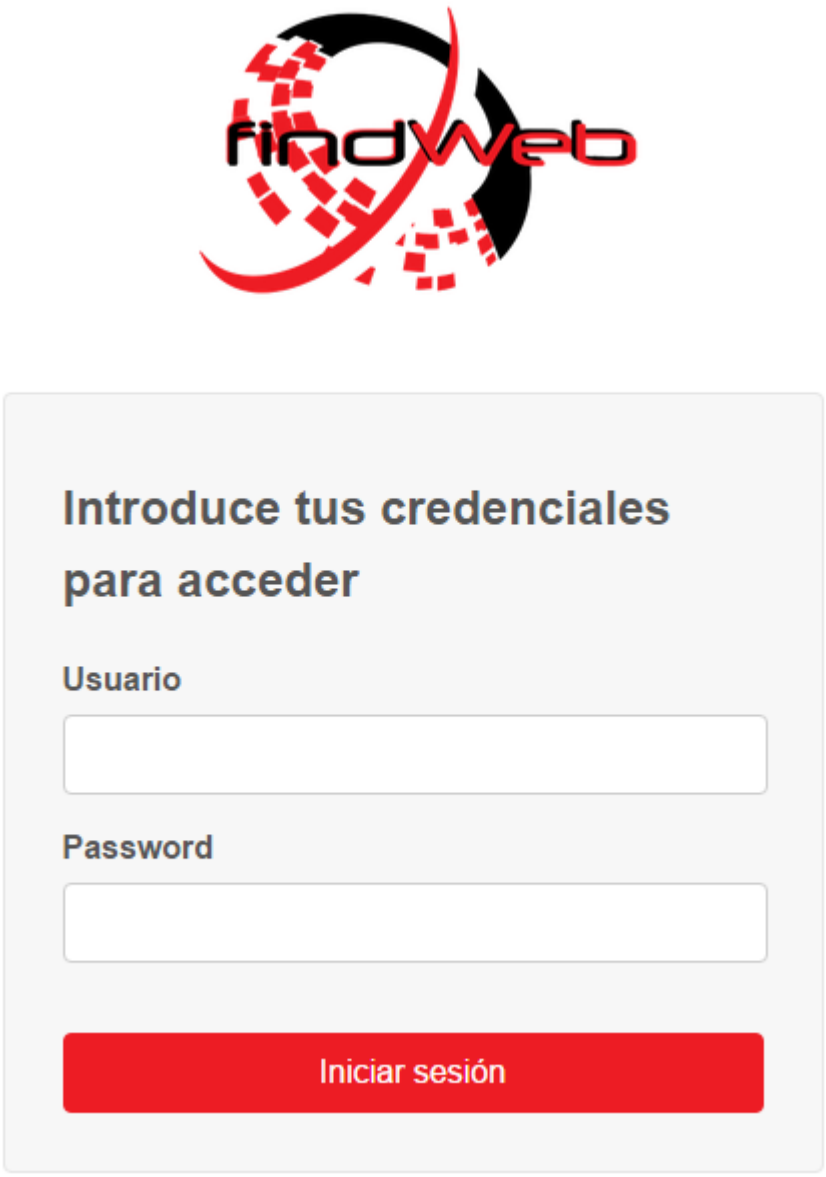

Figura 73. Pantalla de login.

B. Consola web de administración.

1. Partes de la consola de administración:

La consola de administración cuenta con las siguientes partes (ver Figura 74):

- **Menú de administración:** En la parte izquierda de la consola se encuentra el menú donde se encuentran las distintas operaciones que puede realizar el administrador.

- **Botón "Settings":** Este botón permite editar los distintos campos de la cuenta del usuario administrador.

- **Boton "Log Out":** Cierra la sesión actual del usuario administrador.

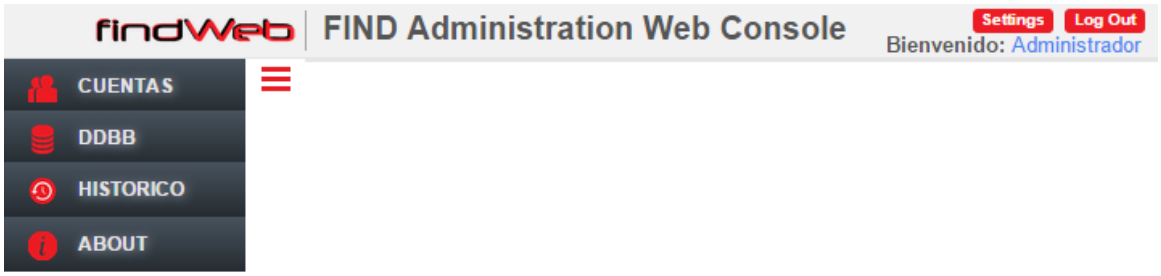

Figura 74. Consola de administración

2. Menú de administración:

I. Para usuarios "FULL ADMIN"

El administrador "FULL ADMIN" tiene acceso completo a las acciones del menú desplegable (ver Figura 75).

Figura 75. Menú disponible para administradores FULL ADMIN

II. Para usuarios "BDD ADMIN"

El administrador "BDD ADMIN" solo tiene acceso a las acciones del menú desplegable que implican operaciones sobre la base de datos Negocio (ver Figura 76).

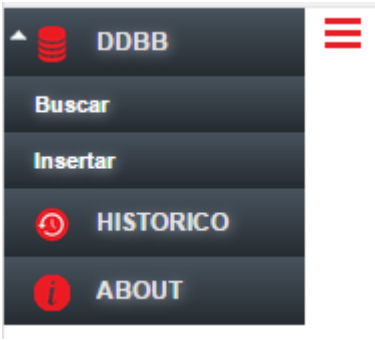

Figura 76. Menú disponible para administradores BDD ADMIN

III. Para usuarios "REPORT ADMIN"

El administrador "REPORT ADMIN" solo tiene acceso a las acciones del menú desplegable que implican realizar operaciones de búsqueda sobre la base de datos Negocio (Ver Figura 77).

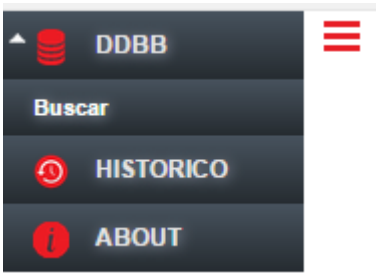

Figura 77. Menú disponible para administradores REPORT ADMIN

3. Botón Settings

En esta pantalla se permite cambiar la configuración de los distintos parámetros de la cuenta (ver Figura 79).

Esta pantalla se encuentra divida en dos partes:

- **Primera parte:** En esta parte se puede editar los campos de Alias, Nombre, Apellidos y Email.

  Para editar los campos es necesario:

  1. **Desbloquear el campo:** Los distintos campos de esta sección se encuentran inicialmente bloqueados. Para poder desbloquearlos

solamente es necesario marcar el "radio button" correspondiente a un campo.

2. **Editar el campo:** Ahora es posible editar el campo.

3. **Salvar los cambios:** Para salvar los cambios realizados en la base de datos solamente es necesario pulsar el botón guardar al final de esta sección.

- **Segunda parte:** Esta parte permite modificar la contraseña del usuario administrador.

  Para editar la contraseña es necesario:

  - El formulario que permite editar la contraseña se encuentra oculto. Para mostrarlo es necesario marcar el "radio button" con la etiqueta "Cambiar Password" (Ver Figura 78).

Figura 78. Formulario de cambio de contraseña.

  - En este formulario se nos pide introducir la contraseña actual una vez y la nueva dos veces.

  - Para Salvar los cambios es necesario pulsar el botón "Guardar" que ha aparecido junto con el resto del formulario.

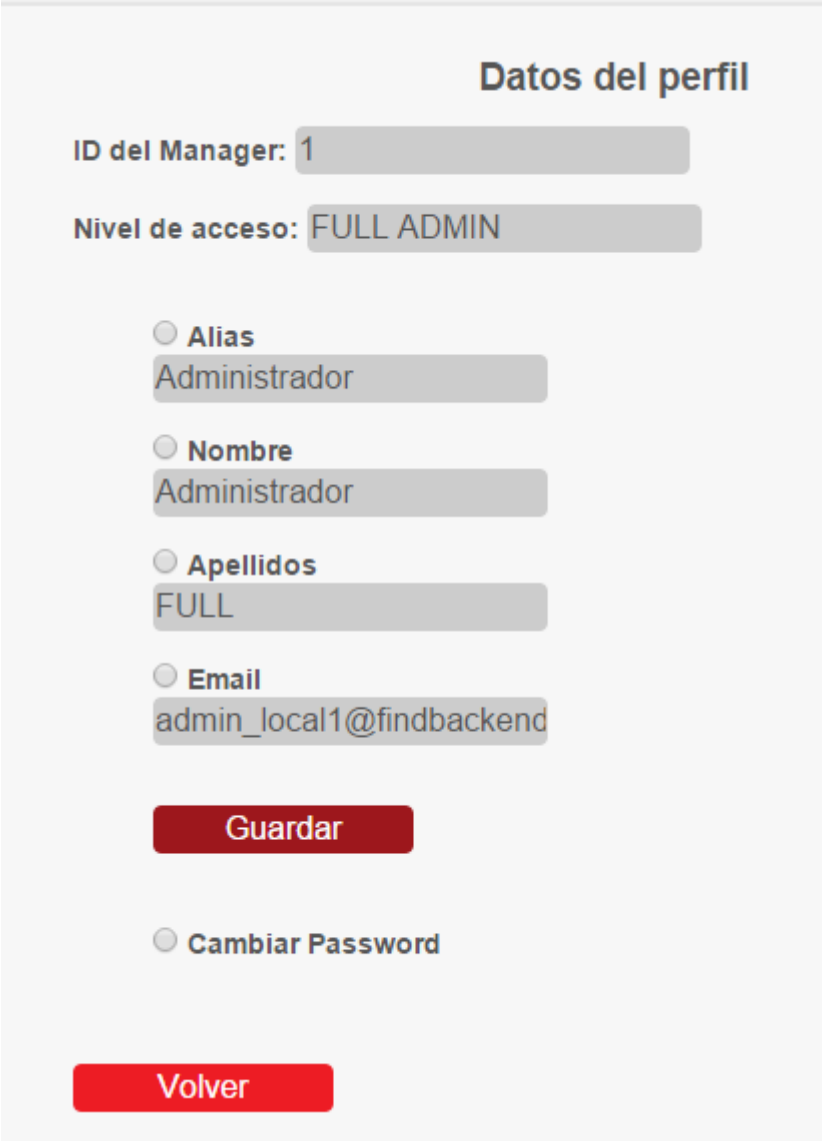

Figura 79. Pantalla Settings

4. Botón "Log Out"

Como se ha comentado el botón "Log Out" cierra la sesión actual y después nos devuelve a la ventana de login.

5. Operaciones del menú:

I. Búsqueda y edición de administradores

Para realizar una búsqueda sobre la base de datos Admin es necesario pulsar en "Cuentas" sobre el menú desplegable, y posteriormente, a la opción "Buscar Admin.

En el formulario de búsqueda (ver Figura 80) introducimos los criterios de búsqueda necesarios y pulsamos buscar. Si dejamos todos los campos en blanco obtendremos todos los usuarios.

Figura 80. Formulario de búsqueda de administradores

Tras esto obtendremos una tabla en el que se mostraran los resultados (Figura 81).

Sobre estos resultados podemos editar los campos de los usuarios administradores. Para poder editar los campos debemos:

- Inicialmente los campos se encuentras desactivados para su edición. Marcar el "checkbox" de la fila que queremos editar. Esto habilitara los botones "Borrar", "Guardar" y "Cambiar_Status", además de los campos correspondientes a la fila marcada.

- Ahora podemos editar los distintos campos del usuario.

- Ahora para guardar los cambios debemos pulsar el botón "Guardar".

  Si lo que queremos es editar el campo "Status", debemos pulsar el botón "Cambiar_Status".

  En cambio, si lo que queremos realmente es borrar el usuario administrador, debemos pulsar el botón Borrar.

Si marcamos variamos "checkbox" al mismo tiempo solo se habilitara el botón "Borrar".

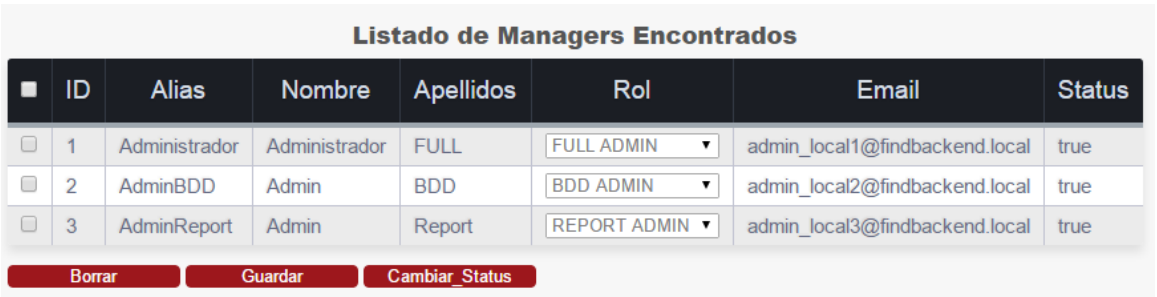

Figura 81. Resultado de una búsqueda

II. Creación de administradores:

Para crear un administrador es necesario pulsar en "Cuentas" sobre el menú desplegable, y posteriormente, a la opción "Nuevo Admin".

Para crear un usuario es necesario rellenar el formulario que aparece en la Figura 82.

Si deja marcado la opción "Enviar correo de activación:" el usuario se creara con el desactivado, y se enviara un correo al email del usuario para que lo active antes de 48 horas.

Figura 82. Formulario de creación de un administrador

III. Búsqueda y edición sobre la base de datos de Negocio

Para crear un administrador es necesario pulsar en "DDBB" sobre el menú desplegable, y posteriormente, a la opción "Buscar".

Antes de obtener los formularios de búsqueda es necesario seleccionar la tabla sobre la que se desea realizar la búsqueda. (Ver Figura 83)

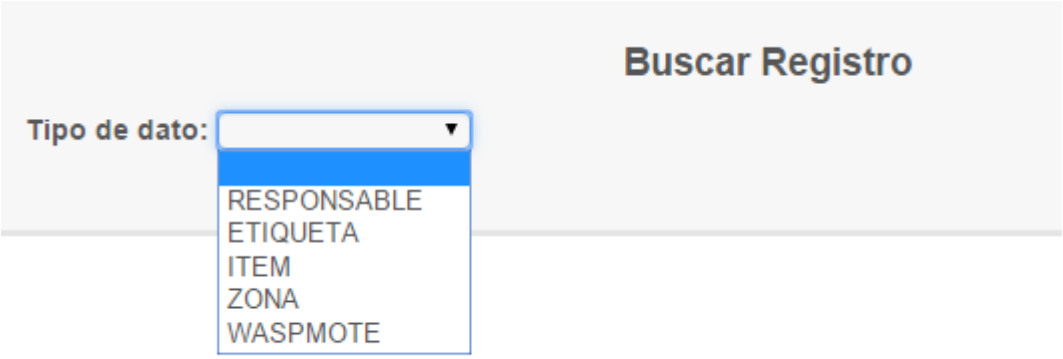

Figura 83. Selector de búsqueda

Puesto que la funcionalidad es similar para cualquier tabla se realizara el ejemplo sobre la tabla responsable.

Para realizar una búsqueda de responsables debemos rellenar el formulario con los datos del responsable que queremos buscar (ver Figura 84). Si dejamos todos los campos en blanco obtendremos la lista de todos los responsables.

Figura 84. Formulario de búsqueda de responsables.

Al igual que en el punto "Búsqueda y edición de administradores" sobre la tabla de resultado es posible editar la información de responsables.

Figura 85. Resultados de la búsqueda de responsables

IV. Inserción sobre la base de datos de Negocio

Para crear un administrador es necesario pulsar en "DDBB" sobre el menú desplegable, y posteriormente, a la opción "Insertar".

Al igual que en el punto "Búsqueda y edición sobre la base de datos de Negocio", antes de obtener los formularios de inserción de datos es necesario seleccionar la tabla sobre la que se desea realizar la inserción. (Ver Figura 86)

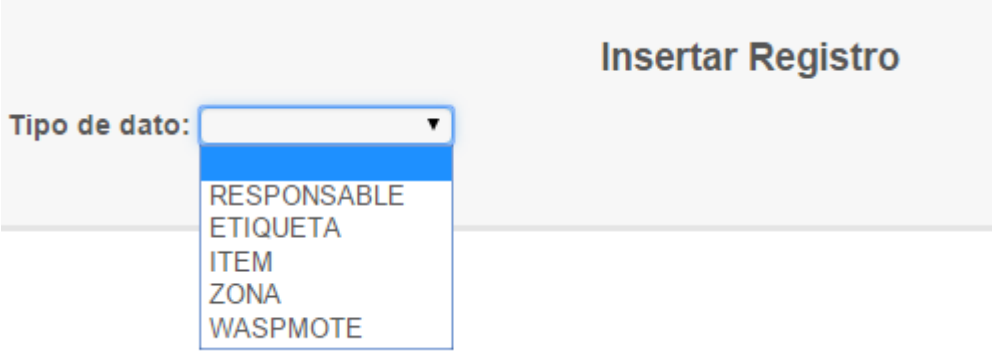

Figura 86. Selector de inserción

Puesto que la funcionalidad es similar para cualquier tabla se realizara el ejemplo sobre la tabla responsable.

Para realizar una inserción de responsables debemos rellenar el formulario con los datos del responsable (ver Figura 87). El sistema no nos dejara realizar la inserción del registro si no se han insertando los campos considerados clave.

Figura 87. Formulario de inserción de responsable

V. Búsqueda de históricos

Para realizar una búsqueda de históricos es necesario pulsar en "Histórico" sobre el menú desplegable.

Para realizar una búsqueda debemos rellenar en el formulario los parámetros de búsqueda deseados (ver Figura 88). De nuevo, si dejamos el formulario en blanco obtendremos todos los históricos.

Figura 88. Formulario de búsqueda de históricos

Capítulo 10

Glosario

10.1 Acrónimos

AJAX	*Asynchronous JavaScript And XML*
API	*Application Programming Interface*
ARPANET	*Advanced Research Projects Agency Network*
CH	*Cluster Head*
CSS	*Cascading Style Sheet*
CSV	*Comma-separated Values*
DARPA	*Defense Advanced Research Projects Agency*
DNS	*Domain Name System*
DSN	*Distributed Sensor Network*
ECMA	*European Computer Manufacturers Association*
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
GNU	*GNU's Not Unix*
GPRS	*General Packet Radio Service*
GSM	*Global System for Mobile Communications*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
IIS	*Internet Information Services*
JDBC	*Java Database Connectivity*
JSP	*Java Server Pages*
JVM	*Java Virtual Machine*
LCD	*Liquid Cristal Display*

LED	*Light-Emitting Diode*
LoRa	*Long Range*
MAC	*Media Access Control*
MVC	*Model View Controller*
NFC	*Near Field Communication*
NTP	*Network Time Protocol*
OTA	*Over The Air*
PDF	*Portable Format Document*
QoS	*Quality of Service*
RFID	*Radio Frequency IDentification*
RTC	*Real Time Clock*
SD	*Secure Digital*
SMTP	*Simple Mail Transfer Protocol*
SOSUS	*Sound Surveillance System*
SPI	*Serial Peripheral Interface*
SQL	*Structured Query Language*
SRAM	*Static Random Access Memory*
SSL	*Secure Socket Layer*
TLS	*Transport Layer Security*
UID	*Unique Identifier*
URL	*Uniform Resource Locator*
WEP	*Wired Equivalent Privacy*
WHATWG	*Web Hypertext Application Technology Working Group*
WLAN	*Wireless Local Area Network*
WPA	*Wi-Fi Protected Access*
WSN	*Wireless Sensor Network*
XHTML	*eXtensible HyperText Markup Language*
XML	*eXtensible Markup Language*

Capítulo 11

Referencias

[1] C.-Y. Chong y S. P. Kumar, «Sensor Networks: Evolution, Opportunities and Challenges,» *PROCEEDINGS OF THE IEEE,* vol. 91, nº 8, pp. 1247-1256, 2003.

[2] Q. Wang y I. Balasingham, «Wireless Sensor Networks - An Introduction,» de *Wireless Sensor Networks: Application-Centric Design, Yen Kheng Tan*, Geoff V Merrett and Yen, 2010.

[3] Silicon Laboratories, «The Evolution of Wireless Sensor Networks,» [En línea]. Available: http://www.silabs.com/Support%20Documents/TechnicalDocs/evolution-of-wireless-sensor-networks.pdf. [Último acceso: Septiembre 2015].

[4] W. Rabiner Heinzelman, A. Chandrakasan y H. Balakrishnan, «Energy-Efficient Communication Protocol for Wireless Microsensor Networks,» de *Energy-Efficient Communication Protocol for Wireless Microsensor Networks*, 2000, 33rd Hawaii International Conference on System Sciences.

[5] I. Urteaga, N. Yu, N. Hubbell y Q. Han, «AWARE: An Activity AWARE network clustering algorithm for Wireless Sensor Networks,» de *IEEE 36th Conference on Local Computer Networks (LCN)*, Bonn, 2011.

[6] S. Bandyopadhyay y E. J. Coyle , «An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks,» *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies,* vol. 3, pp. 1713 - 1723, 2003.

[7] C. Silva, R. Costa, A. Pires, D. Rosário, E. Cerqueira, K. Machado, A. Neto y J. Ueyama, «A Cluster-based Approach to provide Energy-Efficient in WSN,»

International Journal of Computer Science and Network Security, vol. 12, nº 12, pp. 59-66, 2012.

[8] J. N. Al-Karaki y A. E. Kamal, «Routing Techniques in Wireless Sensor Networks: A Survey,» *Wireless Communications, IEEE,* vol. 11, nº 6, pp. 6-28, 2004.

[9] J. V. Capella Hernández, «Redes inalámbricas de sensores: una nueva arquitectura eficiente y robusta basada en jerarquía dinámica de grupos,» Tesis doctoral, Universidad Politécnica de Valencia, 2010.

[10] D. Jain y K. Vijaygopalan, «RFID and Wireless Sensor Networks,» *Proceedings of ASCNT, CDAC,* p. 1 – 11, 2010.

[11] L. Minbo , G. Shengxi , C. Guangyu y Z. Zhu, «A RFID-based Intelligent Warehouse Management System Design and Implementation,» de *IEEE 8th International Conference on e-Business Engineering (ICEBE),* Beijing, 2011.

[12] S. P. Tseng, K. Hwa, I. Chang y W. Li, «An Automatic RFID and Wireless Sensing System on a GHS-based Hazardous Chemicals Management Platform,» de *4th International Conference on Embedded and Multimedia Computing, EM-Com 2009*, Jeju, 2009.

[13] Libelium Comunicaciones Distribuidas S.L., «Waspmote Datasheet,» [En línea]. Available: http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf. [Último acceso: Agosto 2015].

[14] Libelium Comunicaciones Distribuidas S.L., «Waspmote Technical Guide,» [En línea]. Available: http://www.libelium.com/downloads/documentation/waspmote_technical_guide.pdf. [Último acceso: Agosto 2015].

[15] «LoRa Alliance,» [En línea]. Available: https://www.lora-alliance.org/. [Último acceso: Septiembre 2015].

[16] Libelium Comunicaciones Distribuidas S.L., «Waspmote LoRa 868MHz 915MHz SX1272 Networking Guide,» [En línea]. Available: http://www.libelium.com/downloads/documentation/waspmote_lora_868mhz_915mhz_sx1272_networking_guide.pdf. [Último acceso: Agosto 2015].

[17] Libelium Comunicaciones Distribuidas S.L., «Bluetooth Low Energy Networking Guide,» [En línea]. Available: http://www.libelium.com/downloads/documentation/bluetooth-low-energy-networking_guide.pdf. [Último acceso: Agosto 2015].

[18] Libelium Comunicaciones Distribuidas S.L., «Bluetooth for device discovery Networking Guide,» [En línea]. Available: http://www.libelium.com/downloads/documentation/bluetooth-device-networking_guide.pdf. [Último acceso: Septiembre 2015].

[19] Roving Networks, «RN-171 Data Sheet,» [En línea]. Available: http://www.seeedstudio.com/wiki/images/e/ef/WiFly-RN-171.pdf. [Último acceso: Septiembre 2015].

[20] Libelium Comunicaciones Distribuidas S.L., «Waspmote Programming Guide,» [En línea]. Available: http://www.libelium.com/downloads/documentation/waspmote_programming_guide.pdf. [Último acceso: Septiembre 2015].

[21] I. Hickson, R. Berjon, S. Fulkner, T. Leithead, E. Doyle Navarra, E. O'Connor y S. Pfeiffer, «HTML5 A vocabulary and associated APIs for HTML and XHTML,» 28 Octubre 2014. [En línea]. Available: http://www.w3.org/TR/html5/.

[22] «HTML5 Drag&Drop,» [En línea]. Available: http://www.w3schools.com/html/html5_draganddrop.asp. [Último acceso: Septiembre 2015].

[23] «HTML5 Web Workers,» [En línea]. Available: http://www.w3schools.com/html/html5_webworkers.asp. [Último acceso: Septiembre 2015].

[24] «HTML Tutorial,» [En línea]. Available: http://www.w3schools.com/html/default.asp. [Último acceso: Septiembre 2015].

[25] «AJAX Tutorial,» [En línea]. Available: http://www.w3schools.com/ajax/. [Último acceso: Septiembre 2015].

[26] Sun Microsystems Inc., «JavaBeans,» [En línea]. Available: http://download.oracle.com/otn-pub/jcp/7224-javabeans-1.01-fr-spec-oth-JSpec/beans.101.pdf. [Último acceso: Agosto 2015].

[27] Z. Neustupa, R. Danel, P. Stasa, F. Benes y J. Svub, «Ensuring the security of warehouse using automatic identification by RFID,» de *16th International Carpathian Control Conference (ICCC)*, Szilvasvarad, 2015.

[28] Libelium Comunicaciones Distribuidas S.L., «Bluetooth for device discovery Networking Guide..,» [En línea]. Available: http://www.libelium.com/downloads/documentation/bluetooth-device-networking_guide.pdf. [Último acceso: Agosto 2015].

[29] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam y E. Cayirci, «A Survey on Sensor Networks,» *IEEE Communication Magazine,* vol. 40, nº 8, pp. 102-114, 2002.

[30] M. M. Zanjireh y H. Larijani, «A Survey on Centralised and Distributed Clustering Routing Algorithms for WSNs,» de *Vehicular Technology Conference (VTC Spring)*, Glasgow, 2015.

[31] «CSS Tutorial,» [En línea]. Available: http://www.w3schools.com/css/default.asp. [Último acceso: Septiembre 2015].

[32] «JavaScript Tutorial,» [En línea]. Available: http://www.w3schools.com/js/default.asp. [Último acceso: Septiembre 2015].

[33] N. Dimokas, D. Katsaros, L. Tassiulas y Y. Manolopoulos, «High performance, low complexity cooperative caching for wireless sensor networks,» de *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks & Workshops,2009 (WoWMoM 2009)*, Kos, 2009.

[34] Microchip Technology Inc, «Microchip: WiFly Command Reference Manual,» [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/50002230B.pdf. [Último acceso: Septiembre 2015].

[35] Libelium Comunicaciones Distribuidas S.L., «Waspmote Pro API,» [En línea]. Available: https://www.libelium.com/api/waspmote/. [Último acceso: Septiembre 2015].

[36] F. LEWIS, «Wireless SensorNetworks,» de *Smart Environments: Technologies, Protocols, and Application*, New York, D.J. Cook and S.K. Das, 2004, pp. 11-46.